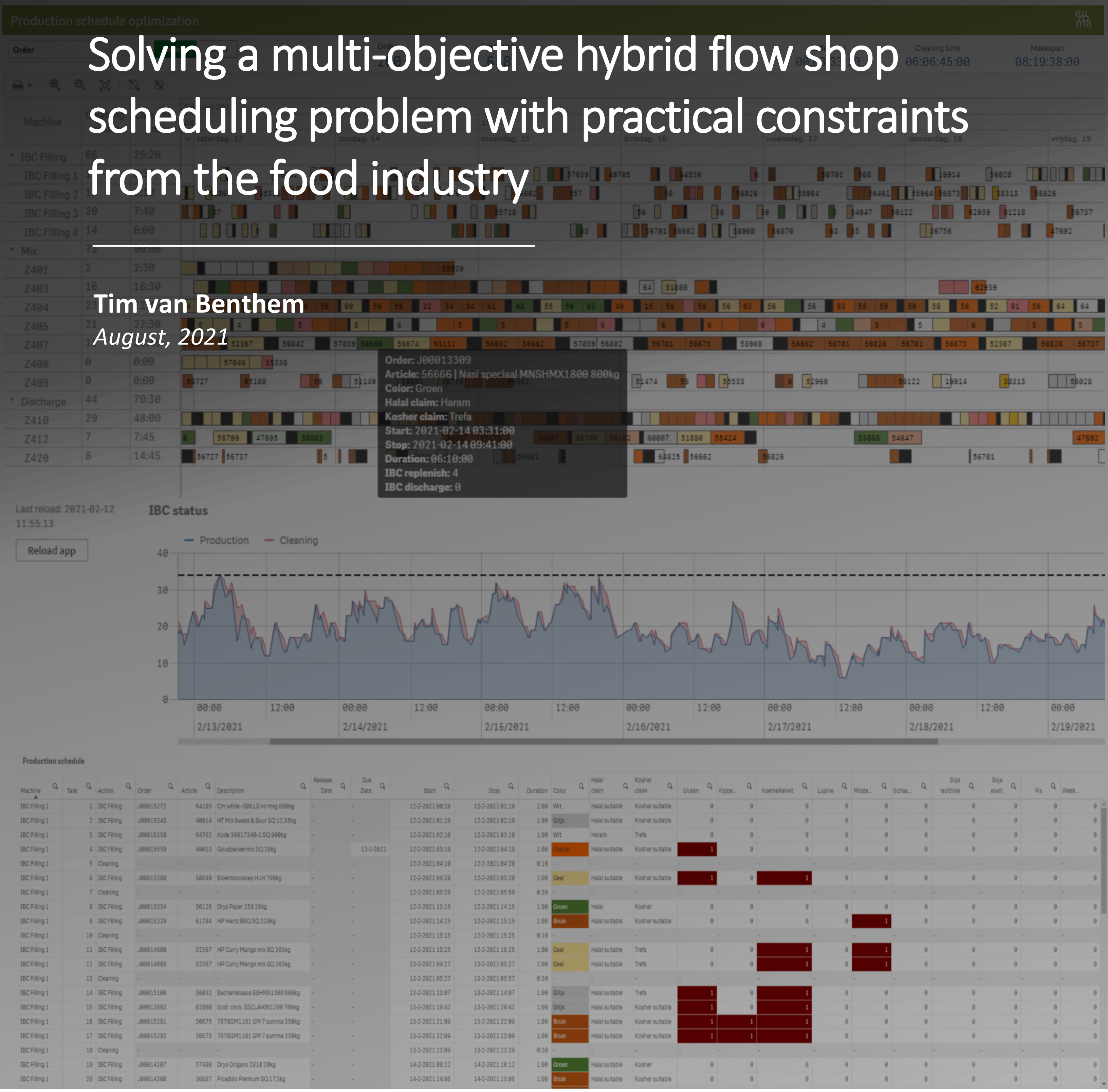
euroma



Solving a multi-objective hybrid flow shop scheduling problem with practical constraints from the food industry

Tim van Benthem

August, 2021

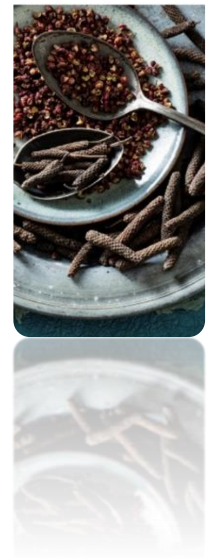
UNIVERSITY OF TWENTE.

united by taste


Solving a multi-objective hybrid flow shop scheduling problem with practical constraints from the food industry

Master Thesis
Industrial Engineering & Management
Production & Logistics Management specialization
University of Twente

August, 2021



Author
Tim van Benthem
MSc Industrial Engineering & Management

Royal Euroma B.V.
Ravensburgstraat 4
8028 PZ, Zwolle
The Netherlands

University of Twente
Drienerlolaan 5
7522 NB, Enschede
The Netherlands

Company supervisors
Mr. J. Vonck
Program Manager at Euroma
Mr. C. Louw
Operating Director at Parcom Capital

University of Twente supervisors
Dr.ir. J.M.J. Schutten
Lead supervisor
Dr.ir. M.R.K. Mes
Supervisor

Preface

This thesis marks the end of my master Industrial Engineering and Management at the University of Twente, where I also completed the bachelor program. During my time as a student, I learned a lot, had many accomplishments and great experiences.

By this means, I would like to take the opportunity to express my gratitude to Euroma, where I was granted the freedom to help to improve the new facility in Zwolle. I experienced great support and willingness to bring this research to a higher level. During this research, I even got the opportunity to implement and test a scheduling model in practice. It was a great experience to see such a large factory producing according to the schedules of a model that I developed by using the knowledge obtained during my master's. This opportunity enhanced my research and the confidence of the organization in decision models.

Furthermore, I would like to express my gratitude to Marco Schutten for providing feedback as my lead-supervisor at the University of Twente. I also would like to thank Martijn Mes for providing valuable input as my second supervisor. Moreover, I would like to thank my fellow student Mark Bergman for his suggestions and for proofreading this report.

Finally, I would like to thank my family and friends for supporting me during this research and helping me move to Zwolle. To this end, I am looking forward to continuing this research project at Euroma in Zwolle, where I will help to implement the proposed scheduling model in practice.

I hope you enjoy reading this thesis and I hope that this thesis contributes to other research as a source of inspiration.

Tim van Benthem
Zwolle, August 2021

Management summary

The facility of Euroma in Zwolle, which produces dry products such as seasonings, herbs, spices, and dry sauces, is at its limits as almost every square meter is occupied. Still, this facility cannot satisfy customer demand. An analysis showed that the mixing department, which consists of three consecutive production stages, is the bottleneck. An OEE analysis identified that the mixers need cleaning 15% of the time and are idle 30% of the time. As a result, the production throughput is less than 300 mixtures per week, whereas 400 mixtures per week are required to satisfy the demand. Therefore, the main research question is:

"How to optimize the multi-stage production schedule to achieve the desired throughput?"

An analysis of the current situation identified the scheduling problem. In essence, the scheduling problem consists of four decisions: (i) allocate an eligible production route to every production job, (ii) allocate an eligible machine to every operation of the job, (iii) determine the sequence of operations on the machines, and (iv) determine the start- and finish times of every operation. Moreover, the problem has many constraints related to, e.g., release dates for jobs, sequence-dependent cleaning times, restricted job sequences, transportation times between stages, a limited number of shared resources, machine maintenance, and production stops.

We conducted a literature review to obtain modeling techniques for this scheduling problem. Based on this review and our insights, we proposed 12 model configurations that each consists of (i) a construction heuristic, (ii) an improvement heuristic, and (iii) a neighborhood structure. Moreover, we proposed a decoding- and a corrective backtracking algorithm to determine the start- and finish times of the operations and cleanings.

We performed experiments to set the weights in the objective function and to obtain the most promising model configuration according to this objective function. Subsequently, we evaluated the following scenarios:

1. Optimize the schedules of the production stages simultaneously instead of separately;
2. Consider all eligible production routes instead of only the default production routes;
3. Optimize the schedules of the production stages simultaneously and consider all eligible production routes instead of optimizing the schedules of the stages separately and only considering the default production routes.

To evaluate these scenarios, we consider 6 problem instances for each scenario from the company data with low-, normal-, and high demand levels. We conducted 25 replications per scenario and problem instance (i.e., solving every scenario and instance 25 times with the same parameter settings to provide statistically significant results). We perform these replications due to the randomness of the heuristics and to obtain the variability of the single objective values in the weighted objective function. We evaluate the KPIs makespan, total tardiness, total cleaning time, average buffer time per job (i.e., the average time that a job waits between the production stages), the number of IBCs needed for the solution, and the percentage of solutions that satisfies the IBC-capacity.

Table 1 provides the average difference per KPI of the three scenarios. The symbols (↑) and (↓) indicate a significant increase or decrease of the KPI with an alpha of 0.01, respectively. The absence of these symbols indicates no significant difference. The colors green and red represent a better and worse performance, respectively.

Table 1 | Average performance difference per scenario

Scenario	Makespan	Tardiness	Cleaning time	Buffer time	IBCs needed	Feasibility
(1) Simultaneous versus separate optimization of production stages	-3.2%	-99.4%	-13.0%	15.9%	6.5%	3.3%
(2) Allowing eligible routes versus only allowing default routes	-16.3%	-33.3%	9.4%	-35.0%	-15.9%	32.7%
(3) Proposed model versus current situation	-19.9%	-100.0%	-1.1%	-37.6%	-12.7%	36.0%

We conclude that optimizing the production stages simultaneously instead of separately significantly reduces the total tardiness. Allowing the model to allocate an eligible production route to a job instead of only considering the default jobs, results in a significant performance improvement for all KPIs, except for the cleaning time; the performance of the cleaning time worsens significantly. This is reasonable since allowing more changeovers results in more flexibility to improve the other KPIs. Improving the other KPIs outweighs the increase in the cleaning time.

Table 2 provides the 99% – confidence intervals (CI) of the KPIs to compare the performance of the current situation with the performance of the proposed model per demand level. The timestamps have the format "d:hh:mm:ss".

Table 2 | 99% – confidence intervals of the current situation and the proposed model

Demand	Model	Current situation	Proposed model	Difference
Low (± 200 jobs)	Makespan	[4:07:04:46 - 4:14:47:38]	[3:15:48:53 - 3:19:28:31]	-16.2%
	Tardiness	[0:01:58:44 - 0:13:21:57]	0:00:00:00	-100.0%
	Cleaning time	[4:08:45:02 - 4:16:10:58]	[3:09:14:33 - 3:17:31:15]	-21.3%
	Buffer time	[0:01:51:14 - 0:02:18:29]	[0:01:58:56 - 0:02:16:41]	2.4%
	IBCs needed	[38.9 – 42.5]	[40.8 – 43.7]	3.7%
	Feasibility	100%	100%	0.0%
Normal (± 300 jobs)	Makespan	[6:05:58:34 - 6:10:54:02]	[4:23:02:47 - 5:02:36:35]	-20.7%
	Tardiness	[0:09:38:30 - 1:09:07:22]	0:00:00:00	-100.0%
	Cleaning time	[6:08:55:16 - 6:14:05:44]	[5:19:28:38 - 6:08:29:10]	-6.1%
	Buffer time	[0:04:16:04 - 0:05:31:53]	[0:03:19:33 - 0:03:52:20]	-26.5%
	IBCs needed	[51.4 – 57.7]	[47.6 – 51.1]	-9.5%
	Feasibility	74%	98%	24.0%
High (± 400 jobs)	Makespan	[8:03:59:24 - 8:05:56:00]	[6:09:52:02 - 6:12:33:42]	-21.2%
	Tardiness	[2:15:17:31 - 6:10:21:32]	0:00:00:00	-100.0%
	Cleaning time	[8:01:40:44 - 8:06:04:16]	[9:04:39:10 - 9:10:14:02]	14.1%
	Buffer time	[0:07:41:56 - 0:08:33:41]	[0:04:03:01 - 0:04:26:44]	-47.8%
	IBCs needed	[67.8 – 72.5]	[50.9 – 54.7]	-24.7%
	Feasibility	8%	92%	84.0%

We conclude that, compared to the current situation, the proposed model significantly improves almost every KPI on every demand level, except for the cleaning time at the high demand level; the cleaning time increases significantly. This is reasonable since allowing more changeovers results in more flexibility to improve the other KPIs. However, having more changeovers may result in a higher cleaning time. Nevertheless, the improvements on the other KPIs outweighs the increase in the cleaning time. An increase in the cleaning time is not a concern since we considered the number of operators that are available for cleaning. Therefore, the capacity of the operators is always satisfied.

All in all, the weekly production throughput, which is the main KPI of Euroma, increases from 300 jobs for the current situation to 400 jobs per week for the proposed model, which meets the desired level of Euroma.

Furthermore, we implemented our model in practice to optimize the mixing schedules by minimizing the total cleaning time and total tardiness. Based on the results, we conclude from the 99% – CIs that the cleaning time reduction is between [26.3% – 49.1%] compared to the situation before implementing the model. This results in a weekly cleaning time saving of the mixers between [0:13:21:14 – 1:00:55:50].

Finally, we recommend Euroma to:

- Implement the scheduling model that can optimize the schedules of the production stages simultaneously and that can allocate production routes to jobs;
- Enrich the input data by logging the processing times of the key production steps to enhance the quality of the solutions of the model;
- Define and monitor a set of KPIs that represent the whole production system, e.g., the IBC flow through the system, the OEE per machine, and the production plan adherence;
- Investigate the costs and consequences for stakeholders and IT systems concerning the implementation of the proposed model.

Table of Contents

List of Figures

List of Tables

1. Introduction

Section 1.1 introduces the company Euroma and Section 1.2 describes the mixing and packaging departments that form the subject of this study. Section 1.3 identifies the core problem of this research. Sections 1.4 and 1.5 introduce the research approach and the research methodology, respectively. Finally, Section 1.6 outlines the structure of this report.

1.1 Introduction to Euroma

Euroma was established in 1899 and started producing herbs and spices in Zwolle. The name Euroma was first used in 1966 and kept on being used from that point in time. In 2001, Euroma was granted the Royal predicate – an acknowledgment of national significance that Euroma occupies an important place in its field (Euroma, History, 2019).

To improve Euroma's market position, Euroma acquired Intertaste in 2018. At that time, Euroma started building her new state-of-the-art production facility in Zwolle. Figure 1 shows the new facility in Zwolle where we conduct our research. This facility has, among others, a robotized high-rise warehouse, automated production lines, automatic guided vehicles (AGVs), and silos that rapidly and automatically supply large volumes of raw materials.

Figure 1 | New production facility in Zwolle

After the acquisition, Euroma had six production facilities and decided to close and merge three of these facilities into the facility in Zwolle. In 2019, the facilities in Utrecht and Puttershoek were closed after the production lines were moved to Zwolle. In 2021, the location in Wapenveld will close, after the production lines have been moved to Zwolle. Figure 2 provides an overview of the facilities.

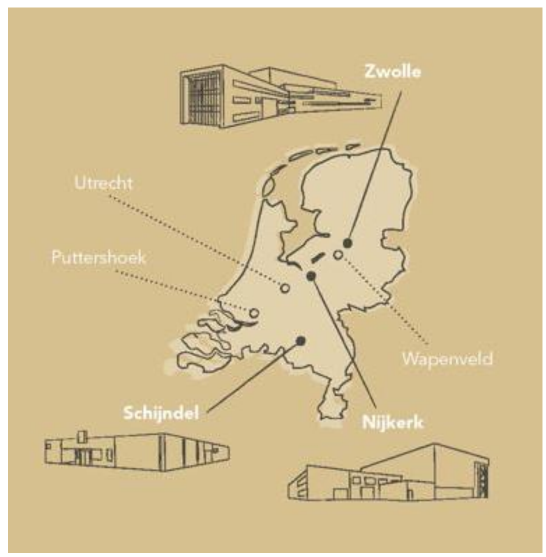


Figure 2 | Production facilities overview

After the merger, Euroma has the following three production facilities: (i) Zwolle, for the production and packaging of dry products such as seasonings, herbs, spices, and dry sauces, (ii) Schijndel, for the production and packaging of ambient liquids, such as mayonnaise and satay sauces, and (iii) Nijkerk, for the production and packaging of fresh liquids, such as dressings and sauces (Euroma, Portfolio, 2019).

At the moment, Euroma has a top 3 position in the European herbs and spices market and a number one position in the Dutch herbs and spices market. Euroma has more than 500 employees and turned over 220 million euros in 2019. Euroma's main mission is to retain a top 3 position in the European herbs and spices market and to deliver her products to all the leading food companies (Euroma, About, 2019).

1.2 Introduction to mixing & packaging

In this section, we introduce the mixing and packaging departments of the production facility in Zwolle.

The new production facility in Zwolle operates twenty-four-seven to produce dry products, e.g., herbal blends, seasonings, coatings, dry (noodle) soups, dry sauces, and instant food. Furthermore, it packages these products for industry and consumer purposes. This facility is highly automized and it produces more than 3000 different end-products. Figure 3 outlines the main processes that are executed in this facility. We briefly explain the replenishing, mixing, and packaging processes.

Figure 3 | Overview of the processes in the mixing and packaging departments

Replenishing

The mixing department has a replenishment system consisting of several silos and different replenishment stations. Here, AGVs transport intermediate bulk containers (IBCs) to collect the ingredients of recipes. Figure 4 shows an example of AGVs, loaded with a 1,500L IBC, driving across different silos to collect pre-weighted ingredients for a recipe.

Figure 4 | AGVs with IBCs collecting ingredients at day silos

Mixing

After collecting the ingredients, the AGVs and the IBC-elevator (i.e., an elevator that is dedicated to IBCs) transport the IBCs to the mixing department. Figure 5 gives an overview of the five-floors mixing department, which consists of different mixers with varying capacities and characteristics. The mixing department produces more than 300 mixtures weekly, each containing tens of ingredients.

Figure 5 | CAD of the mixing department in Zwolle

Figure 6 | Load mixtures into big bags

Figure 7 | AGV transporting an IBC into the IBC-elevator

Packaging

After mixing the ingredients, the mixer discharges the mixture into big bags or into the same IBCs that were used to fill the mixer. Figure 6 shows the discharging of a mixture into two big bags. AGVs transport these big bags and IBCs to the packaging department. Here, packaging lines package the mixtures for industry and consumer purposes. After discharging the IBCs into the packaging lines, the AGVs transport the dirty IBCs to the two manual IBC-cleaning stations. These IBC-cleaning stations can each clean one IBC at a time. Figure 7 shows an example of an AGV that transports an IBC to the IBC-elevator. After finishing the packaging, manual forklift trucks transport the products to the automated high-rise warehouse.

1.3 Problem identification

In this section, we first introduce the problems and illustrate these problems with examples. After that, we create a problem cluster to identify the relationship between these problems. We then select the core problem to solve in our research.

1.3.1 Problem background

After the acquisition, the demand from the facilities in Utrecht and Puttershoek moved to Zwolle to obtain economies of scale. The demand increased significantly and more complex recipes with longer processing times needed to be produced. Initially, only four different liquids were needed, whereas after the acquisition, there are more than 120 different liquids to produce the recipes.

Nowadays in 2021, demand has tripled since 2018 due to the increase in customer demand and the acquisition of Intertaste. To satisfy the demand, 3 more mixers and 10 more IBCs were installed. As a result, the facility in Zwolle is at its limits as almost every square meter is occupied. Figure 8 shows that the facility in Zwolle still cannot satisfy the production requirement to satisfy the customer demand. Therefore, the facility in Wapenveld is still operational, resulting in high additional costs. Note that in week 34, the facility started producing twenty-four-seven. Furthermore, in the weeks 43 and 47 of 2020 new mixers were installed.

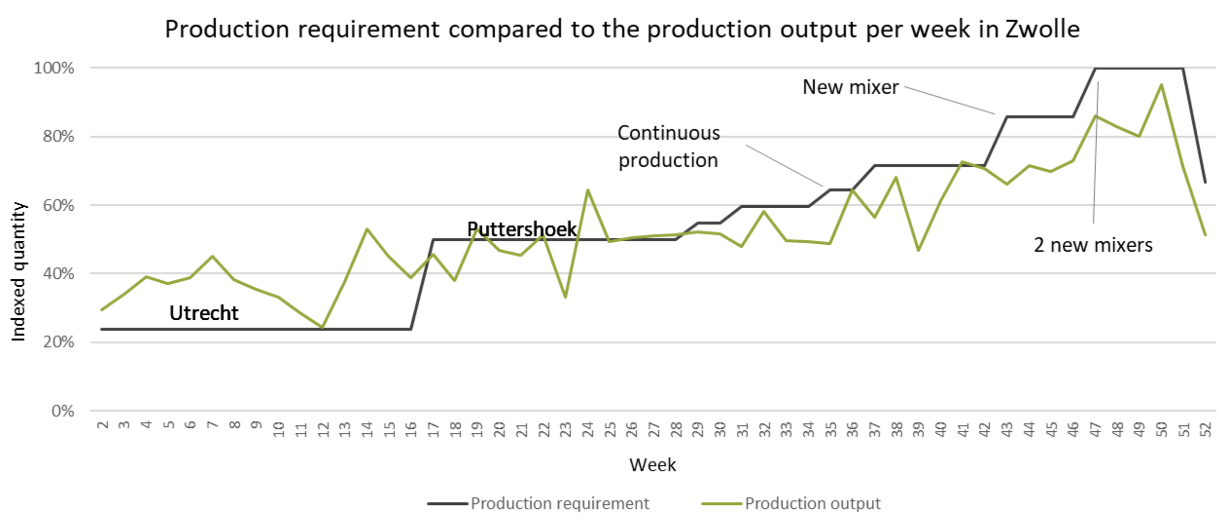


Figure 8 | Production requirement compared to the production throughput per week in Zwolle in 2020

Figure 9 shows that the mixers in the facility in Zwolle currently have an overall equipment effectiveness (OEE) of about 50%. As a result, the production throughput is less than 300 mixtures per week, whereas 400 mixtures per week are required to satisfy the demand.

The mixers have a low OEE since these are often idle due to: (i) changes in the schedule resulting in idle time or extra cleaning time, and (ii) idle time due to waiting on shared resources (e.g., IBCs, AGVs and operators). We illustrate these two situations with examples.

First, the production system is subject to uncertain events that can impact the schedule. Consequently, there are changes in the schedule that frequently result in waiting times. Figure 10 illustrates an example. Consider a scenario where the mixer with a volume of 10,000L, which we refer to as the "10K mixer", is available in one hour. The red job in the schedule needs one hour of replenishment time. Replenishing the raw materials of a new job can be done parallel to the mixing of another job. Thus, the red job gets activated for preparation. However, one of the ingredients of this job suddenly needs inspection. Therefore, it is not possible to produce this mixture at the scheduled time. The yellow job is second-next in the schedule and takes two hours to prepare. Therefore, the mixer is idle for one hour as the mixer must wait for the job to finish replenishing. One possible solution to reduce the idle time would be to sort the jobs on the preparation time. In case the next job only needs one hour and ten minutes to prepare, then the mixer would be idle for only ten minutes.

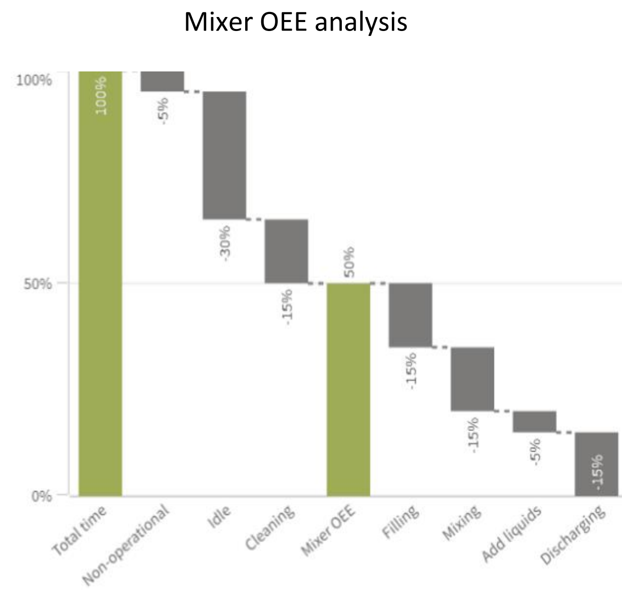


Figure 9 | Mixer OEE analysis (week 35-50 in 2020)

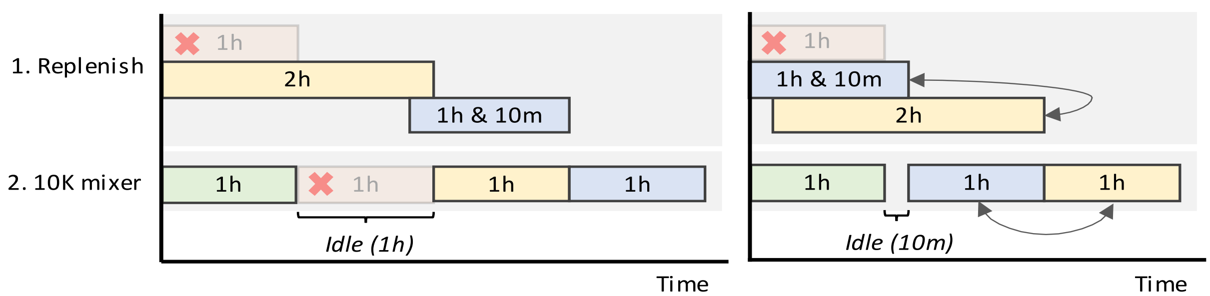


Figure 10 | Robust planning problem example

Second, we provide two examples of situations where machines are idle due to waiting for shared resources. For the first example, consider a situation where multiple mixers need cleaning at the same time. In this situation, multiple operators who are classified to clean mixers are required. The number of operators who are classified for this job is limited. Therefore, mixers become idle due to waiting for operators.

Further, mixing and packaging schedules are currently created manually and separately. First, the mixing schedule is created. Second, based on the mixing schedule, the packaging schedule is created. Often, multiple mixtures that need packaging on the same line finish at the same time. In this case, all IBCs may get occupied. If so, it is not possible to replenish new jobs and fill or discharge mixers. Therefore, the mixers become idle. Also, the other packaging lines can become idle as all the work-in-progress is dedicated to one packaging line.

1.3.2 Problem cluster

To illustrate the relations between the problems that are in the scope of our research, as introduced in Section 1.3.1, we structure the problems in a problem cluster (see Figure 11).

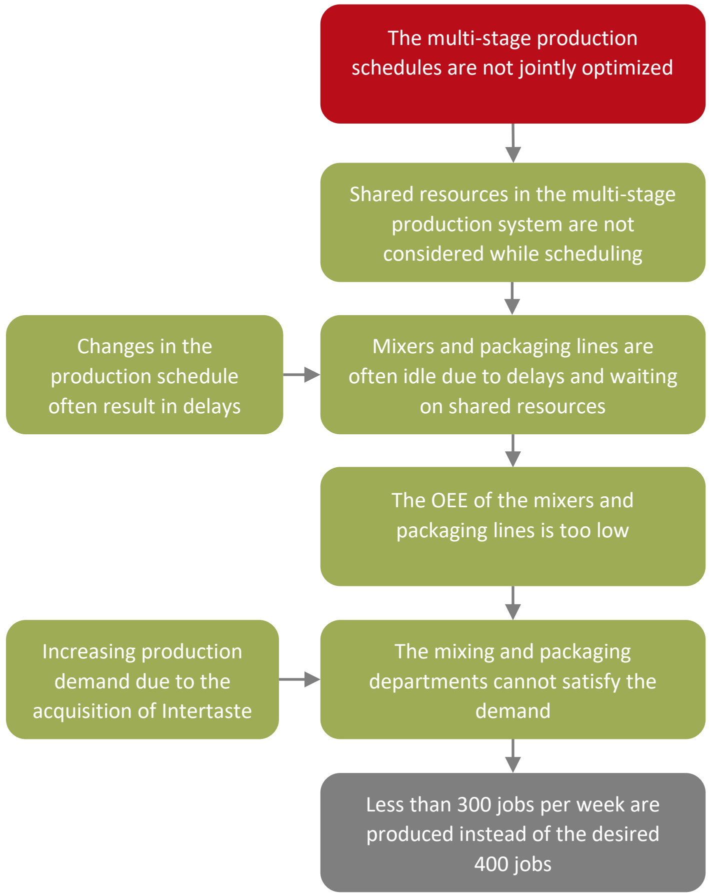


Figure 11 | Problem cluster

Euroma observes the problem at the end of the causal chain in Figure 11. This problem is also referred to as the action problem, which is defined as the discrepancy between the reality and the norm, as perceived by the problem owner (Heerkens & Winden, 2017). The action problem is:

"Less than 300 jobs per week are produced instead of the desired 400 jobs."

To find the causes of the action problem, we investigate and observe the processes. We also interview stakeholders to iteratively discuss and improve the problem cluster until it represents the relations between the problems and their causes. One of these causes, the significant demand increase, is a cause that we do not want to influence. The goal of Euroma is to retain the top-3 position in the European market. To achieve this, demand growth is required.

1.3.3 Core problem selection

Next, we select the core problem from the problem cluster in Figure 11. It is useful to select the core problem, as solving the core problem helps to resolve the action problem (Heerkens & Winden, 2017).

We select the core problem by following the causal chain upstream, starting from the action problem of Euroma. The problems that we cannot influence are not considered to be core problems. The core problem at the end of the causal chain that we can influence is:

"The multi-stage production schedules are not jointly optimized."

We study and solve the core problem in this research. Consequently, we solve the related downstream problems to increase the production throughput.

1.4 Research approach

In this section, we introduce our approach to solve the core problem that we identified in Section 1.3. First, we provide the research objective and we outline the practical- and scientific contribution. Next, we define the scope of the research and then define the research questions of which the answers are required to solve the core problem. Moreover, we also describe the approach to find the answers to the research questions.

1.4.1 Research objective

The main objective of the project is to increase the production throughput. To achieve this, we identified the core problem. Therefore, the objective of the research is to find a method to jointly optimize the multi-stage production scheduling. For this research, we need to take into account the future needs of Euroma, e.g., the increasing customer demand and the additional demand that will be moved from the facility in Wapenveld to the facility in Zwolle.

The research objective regarding the scientific contribution is to investigate whether and to what extent multi-stage production scheduling with practical constraints from the food industry can be jointly optimized. We illustrate the findings using the case study of Euroma.

Regarding the practical contribution, we systematically identify possible improvement points. Further, the objective is to find a method to jointly optimize the multi-stage production scheduling to contribute to the operational performance of Euroma. Therefore, this research aims to contribute to Euroma's mission – retaining a top 3 position in the European herbs and spices market.

1.4.2 Research scope

For this research, we cover relevant and known areas that can be influenced. We consider several crucial production processes of the facility in Zwolle, of which Figure 12 depicts a rough overview.

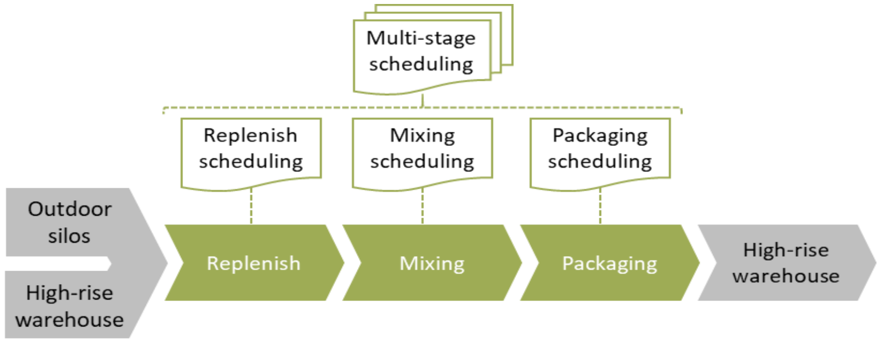


Figure 12 | Research scope process overview

Figure 12 shows in green that the replenishing, mixing, and packaging processes are in the scope of this research. Note that the automated storage process, consisting of the high-rise warehouse and the automated outdoor silos, is not in the scope of this research. These systems simply execute the jobs of the mixing and packaging departments and are not experienced as bottlenecks. Also, these systems operate on confidential third-party software and can therefore not be changed easily.

1.4.3 Research questions & approach

To meet the research objective in a structured manner, we first formulate the main research question. After that, we formulate sub-research questions and describe our approach to find the answers to these questions. The main research question is:

"How to optimize the multi-stage production schedule to achieve the desired throughput?"

The sub-research questions are partitioned into five sections according to the Managerial Problem-Solving Method (MPSM) from Heerkens and Winden (Heerkens & Winden, 2017). Table 1 provides an overview of the MPSM and its application in this research. The structure of the report reflects the structure of the MPSM. Next, we describe the sub-research questions and elaborate on the main content of the report chapters.

Table 1 | Research methodology and application

MPSM methodology		**MPSM application**		
Phase	**Description**	**Question**	**Section**	**Chapter**
1	Define the problem	-	1.3	Introduction
2	Formulate the approach	-	1.4	Introduction
3	Analyze the problem	1	2	Current situation
4	Develop alternative models	2	3	Literature review
		3	4	Model alternatives
5	Select model & evaluate the performance	4	5	Experiments
6	Implement the model	5	6	Model implementation

The first set of sub-research questions aims to analyze the current situation of the multi-stage production processes of Euroma and their performance. We need this information to understand the current situation and to identify improvement opportunities. The sub-research questions are:

1. *What are the current scheduling and production processes in the multi-stage production system?*

Production processes
- What are the current production processes and how are these connected?
- Which machines are in the system and what are the specifications of these machines?
- Which software systems are used in the processes and how do these interact?
- Which data regarding the mixing and packaging processes is available?

Planning and scheduling processes
- What is the process flow from planning to scheduling?
- How are the production schedules currently created?
- What are the objectives and restrictions of the production schedules?

Performance
- How does Euroma currently measure the production performance?
- What is the current production performance?
- What are the possible improvement opportunities?

For answering the sub-research questions regarding the current situation, we use our insights based on the observations, insights from stakeholders, and available data regarding the processes. In case the available data is not sufficient or lacking, we collect the data ourselves. In Chapter 2, we answer these sub-research questions by providing process flow charts, machine specification tables, an overview of the performance, insight into bottlenecks, and possible improvements.

After we have analyzed the current situation, we classify and translate our scheduling problem to theoretical problems available in the scientific literature. We compare our problem to known problems to identify gaps and similarities. Further, we identify suitable models to solve our scheduling problem. Chapter 3 provides a literature review to answer the following sub-research questions:

2. *“Which methods are available in literature for our scheduling problem to increase throughput?”*

Scheduling problem classification and positioning
- Which theoretical scheduling problems are available in the literature?
- How to translate the scheduling problem of Euroma into theoretical problems?
- What are the gaps and similarities between our scheduling problem and problems in literature?

Modeling methods
- How to model complex sequencing- and capacitated resource constraints?
- Which objective functions are often used to increase the production throughput?
- Which neighborhood structures are suitable and what is their connectedness?
- What is the performance of the models available in the literature?

Next, we develop methods to solve the multi-stage production scheduling problem of Euroma. We do this based on literature that we collect from scientific databases and based on the knowledge obtained during the master's in Industrial Engineering and Management. In Chapter 4, we answer the following sub-research questions:

3.	*Which alternative solution approaches are suitable to solve the scheduling problem of Euroma?*
-	Which approaches can deal with complex sequence-dependent constraints?
-	Which approaches take into account limited resources that are used by multiple stages?
-	Which objective is most suitable for the situation of Euroma?
-	Which approaches can solve the problem instances of Euroma in limited computational time?

When we have developed and selected alternative solution approaches, we test these approaches in different experimental settings to analyze the performance. Chapter 5 provides the experimental design including a description of the problem instances, model settings, and experimental results. We analyze the performance and robustness of the solution approaches compared to the current situation. The corresponding sub-research questions are:

4.	*Which alternative solution approach performs best compared to the current situation under different experimental settings?*
-	What production performance can be expected?
-	What is the effect of an increase or a decrease in demand?
-	Which machines are the bottleneck?

Chapter 6 provides information regarding the implementation of the solution approach in practice and highlights the consequences and requirements of the proposed changes. The corresponding research questions are:

5.	*What are the consequences and requirements of the redesigned processes on the system?*
-	What are the IT system requirements?
-	What are the consequences for stakeholders (e.g., production planners and operators)?

Finally, Chapter 7 provides conclusions, recommendations, a discussion, and suggestions for further research.

2. Current situation

The main goal of this chapter is to better understand the current situation and thus the core problem. To achieve this, Section 2.1 provides an overview of the production processes that are in the scope of this research. Accordingly, Section 2.2 describes the planning and scheduling processes that are related to these production processes. Moreover, Section 2.3 describes and illustrates several problems that we observe that are related to the core problem. Besides that, Section 2.4 provides an overview of the IT systems and their main tasks to manage the production processes in this study. Furthermore, as there are many stakeholders involved in this study, Section 2.5 briefly mentions the stakeholders and their perspectives. Finally, Section 2.6 provides a summary of our problem.

2.1 Process overviews

Sections 2.1.1, 2.1.2, and 2.1.3 describe the replenishing, mixing, and packaging processes, respectively. For further reference, Appendix 1 provides the position of these three processes in the general process overview of the facility in Zwolle.

2.1.1 Replenishment process

This section explains the replenishment process according to the process flow in Figure 13. The yellow boxes indicate the replenishment processes. The arrows illustrate the material flows and the colors represent the transportation medium. The dark grey boxes indicate the replenishment processes of small, medium, and large raw material quantities. The remainder of this section explains these three replenishment processes.

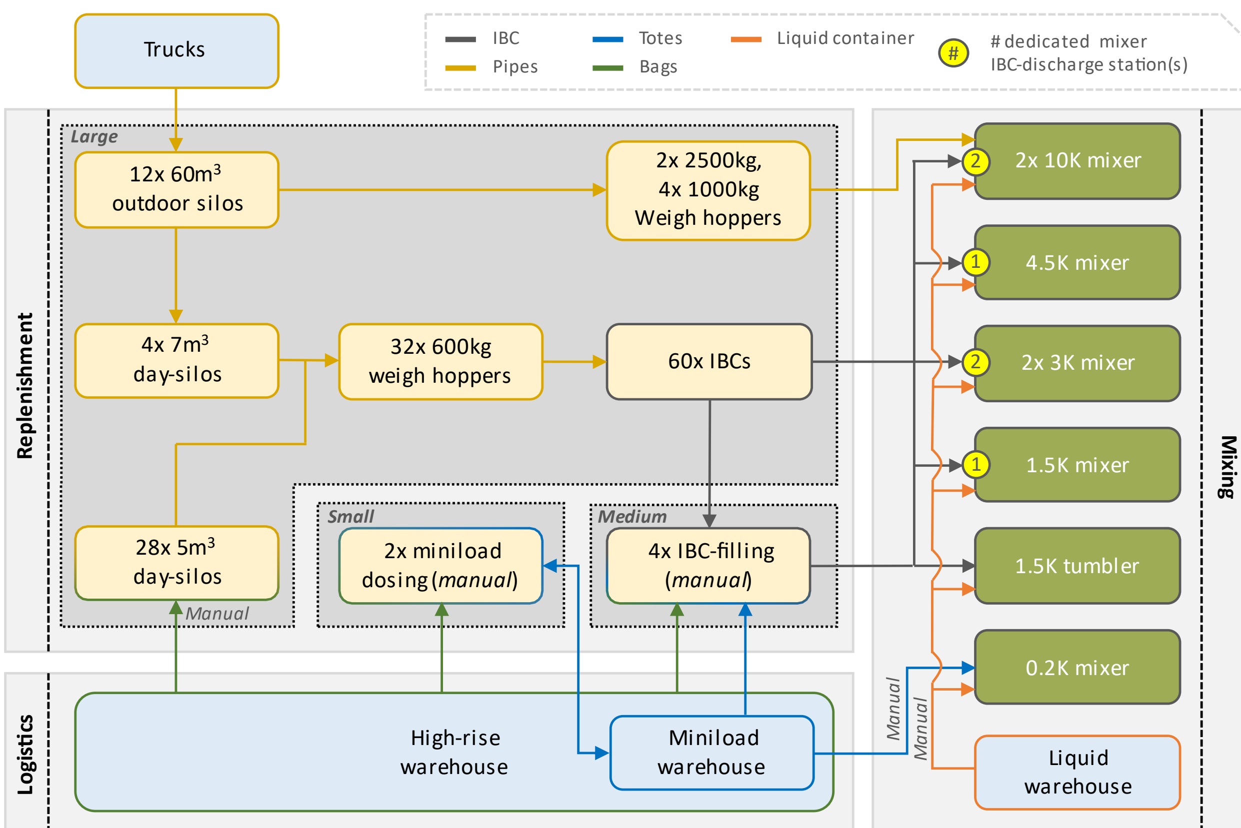


Figure 13 | Replenishing and mixing process flow

First, operators weigh small quantities of raw materials into totes at the miniload stations a few days prior to the mixing process. Small quantities are typically less than 20 kilograms per raw material. The miniload warehouse stores these totes until they are requested by the medium quantity replenishment process.

Second, the medium quantity replenishment process requests the totes with the pre-dosed raw materials from the miniload warehouse, and it also requests pallets holding multiple bags of the same raw material in the range of 10kg – 20kg per bag from the high-rise warehouse. Conveyor belts transport these totes and pallets to four IBC-filling stations. Here, operators discharge a pre-set number of bags into the IBCs manually. Furthermore, operators empty the totes into the IBCs. Note that an IBC-filling station can only replenish the IBCs of one job at a time and a job cannot be split over stations.

Both the IBC-filling stations as well as the miniload stations operate according to the goods-to-man-concept, i.e., all goods come to the operators; the AGVs deliver the IBCs, and the conveyor systems deliver the pallets with bags as well as the totes.

Third, trucks supply the twelve most common raw materials (e.g., wheat flour, salt, or starch) to the twelve outdoor silos, see Figure 14. A pneumatic vacuum piping system automatically transports four of these raw materials from the outdoor silos to four indoor silos (marked in green in Figure 15). The outdoor silos also connect to weighing hoppers (marked in blue in Figure 15). These hoppers weigh the amount needed for a mixture and discharge the weighted raw materials directly into the two 10K mixers.

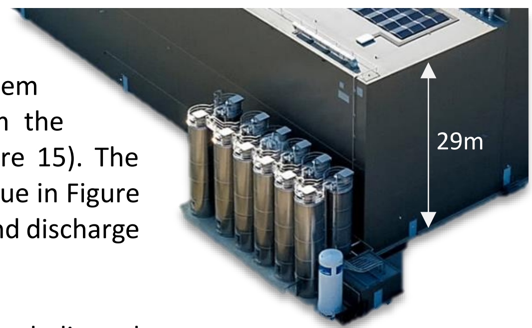


Figure 14 | Outdoor silos

Each of the 32 indoor silos has a weighing hopper and stores a dedicated raw material to avoid cross-contamination of allergens. These indoor silos supply large quantities of raw materials into the IBCs. AGVs, each loaded with a 1,500L IBC, drive to the weighing hoppers to collect the pre-dosed raw materials. After collecting the raw material, the weighing hopper weighs the batch for the next IBC immediately. IBCs that collected all raw materials are placed on a buffer location until they are requested to be discharged into a mixer.

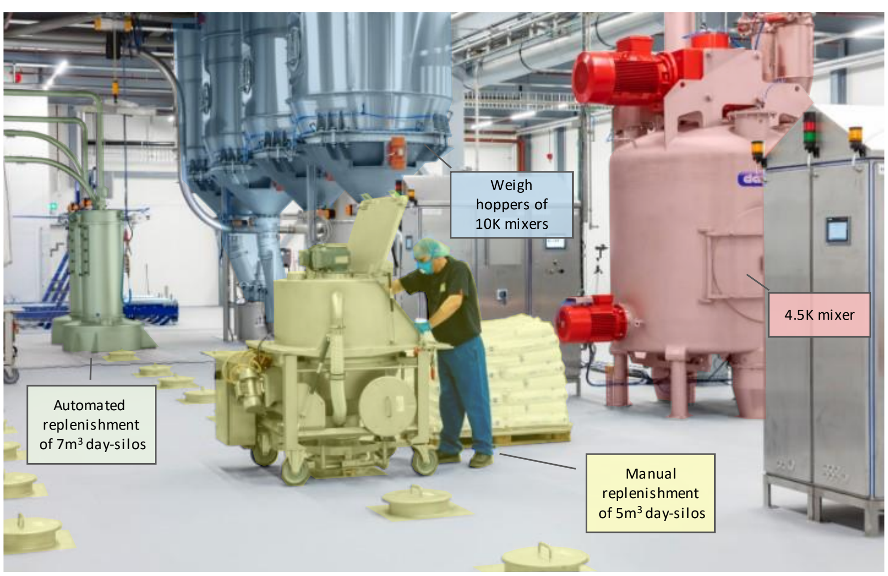


Figure 15 | Replenishment installations

2.1.2 Mixing process

This section describes the mixing process according to the green process stages of the process flow in Figure 13. Moreover, this section explains the filling process of the mixers and the mixer characteristics.

Once all raw materials of a recipe are replenished and the dedicated mixer is ready, i.e., the mixer is idle and clean, the mixing process can start. First, the mixers are filled by the weighing hoppers of the outdoor silos, the IBCs, or by manual replenishment. A combination of the three aforementioned mixer filling methods is also possible. Table 2 provides an overview of the available mixers including their capacity and filling options, where X indicates the possible filling options. Figure 15 displays the 4.5K mixer and Figure 16 shows the mixing blades of the 4.5K mixer.

The weighing hoppers of the outdoor silos can only fill the 10K mixers. This filling process is completely automated by using the vacuum piping system. In case an IBC is required to fill a mixer, the AGVs and the IBC-elevator transport the IBC to the mixer filling station, which is one floor above the mixer. A mixer filling station discharges a single IBC at a time to fill the mixer. This is a sequential process, thus, in case multiple IBCs are required, an AGV puts the IBCs one-by-one on the mixer filling station until all IBCs are empty. The filling of the 0.2K mixer is a manual process in which an operator empties the totes from the miniload warehouse into the mixer. There is one exception, the Tumbler mixer does not need the filling and discharging processes as this mixer can rotate one single IBC, see Figure 17.

Table 2 | Mixer overview and mixer filling options

Mixer			Mixer filling		
Code	Name	Capacity	Outdoor silo	IBC	Manual
Z408	10K	10000L	X	X	Liquid
Z407	10K	10000L	X	X	Liquid
Z404	4.5K	4500L	-	X	Liquid
Z405	3.0K	3000L	-	X	Liquid
Z403	3.0K	3000L	-	X	Liquid
Z402	1.5K	1500L	-	X	Liquid
Z409	Tumbler	1500L	-	X	Liquid
Z401	0.2K	200L	-	-	X / Liquid

Within the mixing department, there is a warehouse that stores about 150 different liquids. In case a liquid is needed for a recipe, an operator weighs the liquid and fills the liquid in the mixer manually.

Figure 16 | Inside of the 4.5K mixer

Figure 17 | Tumbler mixer

2.1.3 Packaging process

This section briefly describes the discharging of the mixers and the packaging process. More details regarding the discharging of the mixers and the packaging process are in Appendix 2.

After the mixing process, the mixers discharge the mixtures via discharging stations that are one floor below the dedicated mixer. Mixers can discharge the mixtures into big-bags, IBCs, or bags. The number of IBCs needed for discharging is less than or equal to the number of IBCs needed for filling the mixer. Most discharging stations have a sieve to filter chunks. In case no sieve is present, the mixtures are for internal use only. Table 3 provides per mixer which discharging station is available.

Table 3 | Mixer discharging stations

Mixers			Mixer discharging stations			
Code	**Name**	**Capacity**	**Big-bag**	**IBC**	**Bag**	**Sieve**
Z408	10K	10000L	X			Yes
Z407	10K	10000L	X			Yes
Z404	4.5K	4500L	X	X		No
Z405	3.0K	3000L	X	X		No
Z403	3.0K	3000L	X	X		No
Z402	1.5K	1500L			X	Yes
Z409	Tumbler	1500L		X		No
Z401	0.2K	200L			X	Yes

After discharging, AGVs or operators transport the mixture to the high-rise warehouse or to a packaging line. Table 4 provides an overview of the input and output of each packaging line. Table 4 also provides the transportation medium to the packaging line. Note that not all mixtures need packaging on a packaging line, e.g., discharging into big-bags is often sufficient when packaging for industry purposes.

Table 4 | Packaging line input and output

Code	Name	Transport	Input	Output
Z410	Votech	AGV	IBC	Bag
Z412	BTH	Operator	Big-bag	Bag
Z420	Dinnissen	AGV	IBC	Big-bag

A sequence of suitable production machines for a job is referred to as a production route. Note that machines can only process one job at a time and preemption of jobs on machines is not allowed. For an example of a set of production routes, consider a job that is suitable for mixing on the 1.5K and both 3.0K mixers. This job needs final packaging in bags. Table 5 lists five suitable routes for this job. Note that route 5 does not need a packaging line as the 1.5K mixer discharges in bags and has a sieve. For further reference, Appendix 3 provides technical information regarding production routes.

Table 5 | An example of suitable production routes

Route	Mixer	Discharging station	Packaging line	Packaging unit
1	Z403 (3.0K)	IBC	Votech	Bag
2	Z403 (3.0K)	Big-bag	BTH	Bag
3	Z405 (3.0K)	IBC	Votech	Bag
4	Z405 (3.0K)	Big-bag	BTH	Bag
5	Z402 (1.5K)	Bag	-	Bag

2.2 Planning and scheduling overview

This section describes the production planning and scheduling processes of the mixing and packaging departments. These processes can be partitioned into three levels. Table 6 outlines per level who is responsible, the horizon, and the corresponding tasks. Figure 18 provides a schematic overview of the planning and scheduling process.

Table 6 | Planning and scheduling levels

Level	Who	Horizon	Task	Job status
Tactical	MRP planners	Monthly / Weekly	Demand planning Batching Allocate routes to jobs	Plan
Offline operational	Operations manager	Weekly / Daily	Re-assign routes to jobs Sequencing	Release
Online operational	Control room operators	Continuous	Determine job release time	Activate

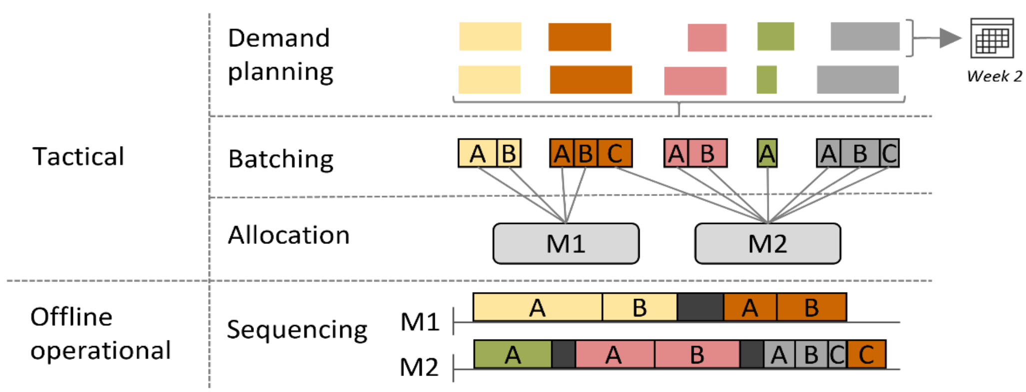


Figure 18 | Schematic overview of the planning and scheduling process

At the tactical level, the Material Requirements Planning (MRP) planners construct week-plans by dividing demand over weeks. Unfortunately, there is little similarity between demand patterns over the weeks. Therefore, the MRP planners divide the demand into production batches. These batches are referred to as jobs. The planners allocate the jobs to production routes and set the job status to "planned". Based on this week-plan, the operations manager sequences a daily offline operational production schedule for the mixers and sets the job status to "released". Once a job is released, the allocated production route is fixed. The control room operators execute the production sequence and set the job status to "active". Once the status of a job is active, production starts and resequencing is not possible anymore for this job. The control room operators manage the online operational interventions that affect the schedule. Based on the status of the system, they determine when to activate jobs. The aforementioned levels are not standalone as there are upstream and downstream interactions. For example, the operations manager can suggest changing the route of a job, which affects both the tactical as well as the online operational levels.

Next, Sections 2.2.1, 2.2.2, and 2.2.3 describe the planning and scheduling processes on the tactical-, offline operational- and online operational levels, respectively.

2.2.1 Tactical week plan

At the tactical level, the MRP planners manually divide all demand over the weeks. To achieve this, the MRP planners use the information provided by the Enterprise Resource Planning (ERP) system. For constructing the week-plans, the planners consider the job release- and due dates and the eligible production routes. Release dates are necessary to ensure that jobs only start production when all raw materials are available. The due dates are required to ensure that customers receive their jobs in time.

The default production route of a job is the route that the ERP system suggests according to the input of a process engineer. These routes include, amongst others, the allocation of jobs to mixers and packaging lines. Generally, the MRP planners allocate jobs to the default production routes. Next, the planners determine the workload per mixer. However, the processing times of the products on the machines are currently unknown. Therefore, the planners estimate the workload per mixer based on the number of allocated jobs and the average achieved number of jobs per shift over the last six weeks. In case this workload exceeds the capacity of a mixer, the planners re-allocate jobs to non-default routes. As the planners only consider the allocated workload and capacities of the mixers, the workloads of the packaging lines are neglected.

The MRP planners present the tactical week-plan to the operations manager every Thursday. During this meeting, the operations manager estimates whether it is possible to realize the new week-plan. The planners need to reconsider the week-plan in case the operations manager foresees problems.

2.2.2 Offline operational schedule

This section first explains some scheduling constraints. After that, this section describes how the operations manager constructs an offline operational production schedule.

On IBC-filling stations, mixers, and packaging lines, cleaning between two consecutive jobs on the same machine is required when at least one of the following three conditions is applicable: (i) when producing a non-allergen product after an allergen product (containing, e.g., gluten, eggs, or sesame), (ii) when colors of two consecutive products can blend into another color, and (iii) when the raw materials of two products have different physical characteristics (e.g., aroma, particles structure, or stickiness).

There are two cleaning types: dry-cleaning and wet-cleaning. Wet-cleaning takes longer than dry-cleaning, as this cleaning type is more intensive. The cleaning durations also depend on the machines. However, the cleaning durations are currently unknown. Wet-cleaning is always required when producing a non-allergen product after an allergen product. Wet-cleaning is in some cases also required based on the colors and physical characteristics of the products. For example, wet-cleaning is necessary when producing a white product after a red product, as the white product can blend into pink in case some red product remains in the machine. Dry-cleaning is only suitable between two consecutive jobs on the same machine in case there is no cleaning required based on allergens. For example, dry-cleaning is sufficient when mixing a yellow product after an orange product, as these colors are somewhat similar.

Moreover, products can have certain claims, e.g., halal, kosher, vegan, or bio. The claims of a product always belong to one of the following three categories: non-suitable, suitable, or certified. For example, consider the halal claim. In this case, a product can be haram (i.e., non-suitable according to Islamic dietary laws), halal-suitable (i.e., suitable according to Islamic dietary laws), or halal (i.e., certified according to Islamic dietary laws). When scheduling a product that is certified for a claim, the two preceding products on the same machine should be suitable or certified for that claim. This constraint is necessary to ensure that the remaining raw materials are flushed out of the piping system before producing a certified product.

Based on the week-plan, the operations manager manually constructs daily mixing schedules in Excel. The operations manager determines the production sequence of the jobs on the mixers. To achieve this, the cleaning requirements and the claims of the products are taken into account. Besides that, release dates and due dates of jobs are considered. The objective of these daily schedules is twofold: (i) minimizing the number of cleanings and (ii) minimizing the job tardiness. These objectives can conflict; in that case, the operations manager decides based on experience which schedule is most suitable.

Next, as the facility is producing continuously, the operations manager takes into account the last jobs per machine of the previous production schedule. The finish times of the machines of the previous schedule are referred to as the machine release times of the new schedule. The first operation of the new production schedule cannot start earlier than the machine release time, otherwise the new schedule conflicts with the previous schedule. Besides that, some machines may have production-stops, e.g., during holidays or audits. Moreover, the operations manager occasionally needs to schedule maintenance activities for machines. Often, the start of these maintenance activities is somewhat flexible, as Euroma has an in-house maintenance service team.

After scheduling, the operations manager estimates whether it is possible to realize the production schedule. At this stage, the operations manager estimates the workload compared to the capacity per mixer and packaging line. This is necessary since the MRP planners do not consider cleaning time. In case there are foreseen capacity problems, rescheduling is needed. The operations manager reschedules iteratively until all foreseen problems are managed.

2.2.3 Online operational scheduling

The control room operators execute the daily production schedule provided by the operations manager. To achieve this, the control room operators manually decide, based on experience, when to release process steps of a job by taking into account the current status of the system (e.g., production progress, or IBC availability). Moreover, they manage interventions in the system (e.g., breakdowns). In case the production schedule cannot be met, the operations manager reschedules the jobs accordingly.

2.3 Current scheduling problems

This section describes and illustrates several problems of Euroma regarding scheduling that we solve in this study. At first, Section 2.3.1 describes the problem of the current scheduling objective. Section 2.3.2 describes the allocation logic of jobs to IBC-filling stations and the problem thereof. Moreover, Section 2.3.3 explains why it is important to take into account the effect of the different production routes. Finally, Section 2.3.4 describes the importance of considering the limited number of IBCs while scheduling.

2.3.1 Current objective

There are two cleaning types: dry-cleaning and wet-cleaning. Wet-cleaning takes longer than dry-cleaning, as this cleaning type is more intensive, see Section 2.2.2. The operations manager does not take into account the cleaning time; the objective is simply to minimize the number of cleanings. The problem with this objective is that the total cleaning time is not minimized. For example, Table 7 provides a scenario where dry-cleaning takes 30 minutes and wet-cleaning takes 60 minutes. In this scenario, there are two feasible schedules: (1) with 3 cleanings and 150 minutes of total cleaning time, and (2) with 4 cleanings and 120 minutes of total cleaning time. When minimizing the number of cleanings, one would opt for schedule 1. However, schedule 2 requires less total cleaning time.

Table 7 | Cleaning objective scenario

Schedule	Type	Count	Time	Total time
1	Dry	1	30	30
	Wet	2	60	120
	Total	3		150
2	Dry	4	30	120
	Wet	0	60	0
	Total	4		120

Moreover, we question whether the current objective (i.e., minimizing the number of cleanings and the job tardiness) ensures that production finishes as soon as possible. This is necessary such that capacity remains available for future jobs, as the facility is producing twenty-four-seven.

2.3.2 IBC-filling stations allocation

Furthermore, the control room operators experience IBC-filling stations as a bottleneck. They state that mixers are often idle due to waiting for IBCs to replenish at the IBC-filling stations. When analyzing the OEE data of the IBC-filling stations over 2020 (see Figure 19), we notice that a significant amount of time is due to idleness and cleaning.

When observing the IBC-filling stations, we notice a queue of jobs at some stations, whereas other stations are idle. This phenomenon results from the logic of allocating jobs to IBC-filling stations. When executing the offline operational production schedule, the process control system ESA (see Section 2.4.1) automatically allocates jobs for replenishment to the four IBC-filling stations. ESA assigns a job to the station that has the least number of jobs in the queue. In case there are multiple stations with the least number of jobs in the queue, ESA assigns jobs to stations in ascending order. ESA does not consider the expected processing time of the jobs. For example, consider a scenario where every station has one job in the queue. The jobs of the first three stations need one hour of processing time and the job of the last station needs ten minutes of processing time. In this scenario, ESA assigns the next job sequentially to the first station. Thus, the last station becomes idle after ten minutes, whereas there is a queue at the first station.

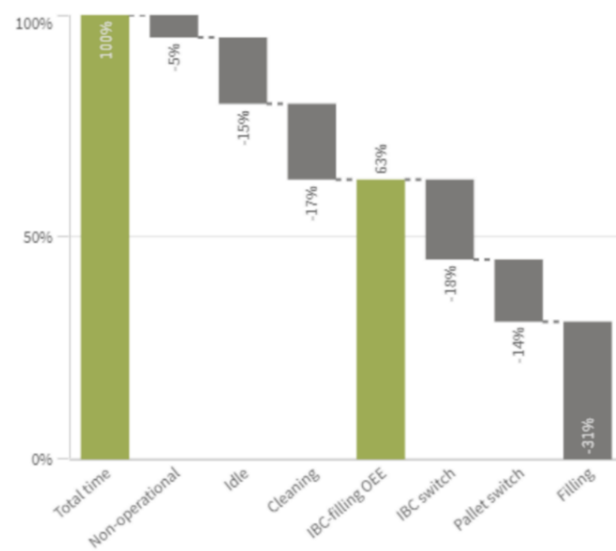


Figure 19 | OEE analysis of the IBC-filling stations in 2020

Another consequence of this allocation logic is that the sequence of jobs on an IBC-filling station is independent of the product characteristics (e.g., color and allergens). Therefore, cleaning the IBC-filling stations is necessary after processing every job to avoid cross-contamination. When assigning jobs with the same product characteristics to the same IBC-filling station, less cleaning is required.

2.3.3 Production route allocation

Section 2.2.1, describes that the MRP planners generally only consider the default production route. As a result, the planners do not consider the allocated workload to the packaging lines. A similar principle holds for the allocated workload to the IBC-filling stations, as this also results from the allocation of jobs to production routes. For example, consider a scenario where it is possible to assign a job to a production route with the 10K mixer or the 4.5K mixer. The 10K mixer can replenish directly via the outdoor silos as well as via IBCs, whereas the 4.5K mixer can only replenish via IBCs. When assigning the job to the 10K mixer, replenishing one IBC is required for this job; the outdoor silos replenish the remaining raw materials. The 4.5k mixer, however, needs four IBCs for replenishment. Replenishing one IBC takes on average 40 minutes. Thus, when allocating the job to the 10K mixer, the IBC-filling station needs 40 minutes of processing time, whereas the 4.5K mixer needs 2 hours and 40 minutes of processing time. Therefore, the workload of the IBC-filling stations is dependent on the allocation of jobs to production routes. As the MRP planners generally do not consider other production routes than the default option, the workload of the IBC-filling stations is not considered. Therefore, considering other production routes is important, especially since the IBC-filling stations are experienced as bottlenecks, see Section 2.3.2.

2.3.4 Limited IBCs

When observing the manufacturing process, we notice that mixers are often idle as there are no IBCs to replenish the mixers. We develop a dashboard to analyze this phenomenon. Figure 20 shows the dashboard where the Gantt chart visualizes the jobs on the machines and the stacked bar chart visualizes the status of the IBCs over time. In this scenario, we notice from the bar chart that there are no clean IBCs available (green bars) between 00:00 and 03:00. Consequently, the mixers become idle as there are no clean IBCs to replenish the mixers. Between 03:00 and 04:00, the operators start cleaning the dirty IBCs, and at 04:30, the mixers become operational.

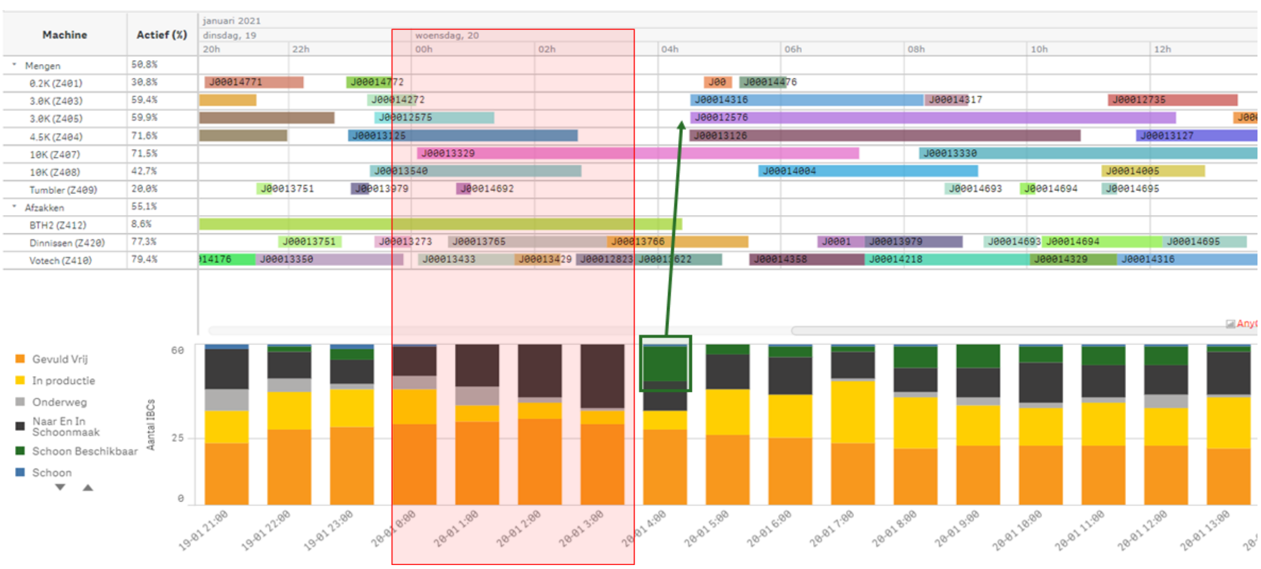


Figure 20 | IBC-status dashboard

The phenomenon of having idle mixers as there are no IBCs to replenish the mixers also occurs when all IBCs are waiting on buffer locations to be discharged into packaging lines. This happens when all jobs need to be discharged at the same packaging line. Therefore, it is crucial to take into account the limited number of IBCs while scheduling. Moreover, as the facility is producing continuously, there can be dirty IBCs at the cleaning station at the start of a new production schedule.

2.4 IT systems overview

Sections 2.4.1 and 2.4.2 explain the IT systems involved in the mixing and packaging and the communication between these systems, respectively.

2.4.1 IT systems

Infor LN is the ERP system of the facility in Zwolle. This system contains the material resource planning, job information, and some product information (e.g., product claims, colors, and production routes). The ERP system also stores the week-plans of the mixing department. The MRP planners manually store the offline operational schedules from the Excel files of the operations manager into the ERP system. This process is time-consuming as every job takes about one minute to import, and there are more than 300 jobs per week.

The product development and the food specification departments mainly use the software PLS Pro. PLS Pro stores, amongst others, the allergen information of the products.

ESA is the process control system of the mixing department. ESA manages, creates, and tracks all the processes in the mixing department (e.g., dosing, IBC transport, mixing, or discharging). The operators in the control room mainly work with ESA, as this system has the most control over all the automated processes. Objective is the manufacturing execution system (MES) of the packaging department. Objective manages and tracks all the processes in the packing department. Furthermore, the software DS Automation manages the AGV transport.

2.4.2 IT systems communication

Figure 21 provides a rough overview of the communication between the IT systems. From top to bottom, the ERP system extracts allergen information from PLS Pro. The MRP planners manually trigger the communication of the mixing jobs from the ERP system to ESA by setting the job status from “planned” to “released”. The MRP planners also manually trigger the communication of the packaging job to Objective. ESA and Objective both communicate transport jobs for the AGVs to DS Automation. Both ESA and MES communicate with the programmable logic controllers (PLCs). Lastly, IQBS is a reporting tool that combines data from the ERP, ESA, and MES.

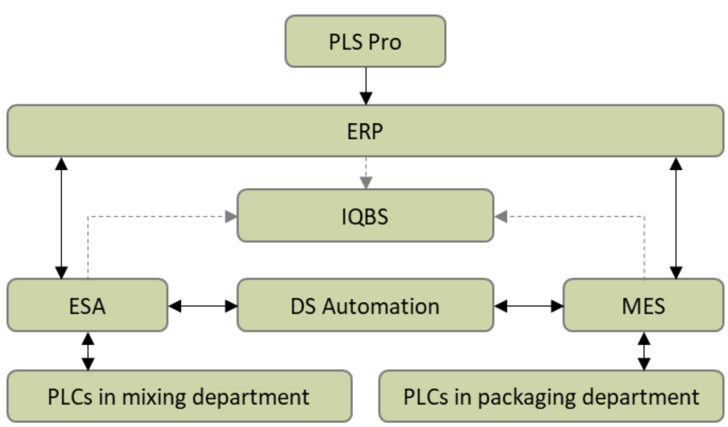


Figure 21 | IT systems communication overview

2.5 Stakeholders

There are many stakeholders involved in the production scheduling process of the mixing and packaging departments. We briefly mention their interests and perspectives. Note that the research perspectives of this study are based on the viewpoints of the stakeholders. We adopt these viewpoints as the angles of view from which we observe phenomena and set objectives during the conduct of this research.

First, Euroma aims to retain its top-3 position in the European herbs and spices market. Currently, the facility in Wapenveld is still operational due to limited production throughput in Zwolle. In case the production throughput in Zwolle increases to the desired level, the facility in Wapenveld can close. The (operations) management aims to close the facility in Wapenveld as soon as possible. Therefore, they need to increase the production throughput in Zwolle.

Next, the MRP planners determine which jobs to produce in the coming weeks, taking into account the production capacity and customer requirements. For the near future, it is determined that the MRP planners should create the production schedules instead of the operations manager. Thus, the MRP planners need to obtain knowledge about production scheduling.

The control room operators execute the production schedules. In case of any unforeseen events that could disrupt the schedule, the control room operators need to be able to reconstruct a good schedule in a limited time, as the MRP planners are not available at any time.

Next, adding more functionality and workload to an IT system often requires more maintenance, so the IT department and the IT software suppliers are crucial stakeholders as these departments need to embed the production scheduling logic in the IT environment. It also requires a lot of data (e.g., job- and product information, processing times, and production routes). Such large datasets often require maintenance, so the data management department should embed these data flows into their systems.

2.6 Summary of the problem

This section summarizes the scheduling problem that Chapter 1 identifies and Chapter 2 analyzes.

The production process has multiple stages: IBC-filling, mixing, and packaging. Jobs can have multiple operations that need processing in a predetermined order (e.g., IBC-filling, mixing, or packaging). To solve the multi-stage scheduling problem, the model needs to (i) allocate jobs to production routes, (ii) allocate operations of production routes to machines, and (iii) sequence the operations of jobs on the allocated machines.

Currently, the scheduling objective is twofold: (i) minimizing the number of cleanings and (ii) minimizing the job tardiness. However, Section 2.3.1 describes why this objective is not suitable for Euroma. Moreover, we question whether this objective ensures that production finishes as soon as possible such that capacity remains available for future jobs, as the facility is producing twenty-four-seven. Therefore, the model needs to have an objective that is most suitable to increase the production throughput while satisfying customer needs.

The scheduling problem is restricted by several constraints. We classify these constraints into the categories scheduling-, sequencing-, and additional shared resource constraints. The constraints of these three categories are listed, respectively.

Scheduling constraints

- The operations of a job can only be processed on one machine at a time (see Section 2.1).
- Machines can only produce one operation at a time (see Section 2.1.3).
- Preemption of operations on machines is not allowed (see Section 2.1.3).
- The production route is fixed for “released” jobs (see Section 2.2).
- The production schedule is fixed for “active” operations of jobs (see Section 2.2).
- Machines cannot produce during production-stops, e.g., during a holiday (see Section 2.2.2).
- Machines cannot produce during maintenance (see Section 2.2.2).

Sequencing constraints

- The operations of a job have a predetermined sequence (see Section 2.1.3).
- Cleaning times are sequence-dependent (see Section 2.2.2).
- There are fixed transportation times between consecutive operations of a job (see Section 2.1).
- Jobs and maintenance can only start after the release date. Jobs and maintenance should finish before the due date, which is a soft constraint (see Section 2.2.2).
- The job sequence on a machine is constrained (e.g., the claims as Section 2.2.2 describes).
- The new schedule should comply with the previous schedule (see Section 2.2.2).
- Operations on a machine cannot start earlier than the machine release time (see Section 2.2.2).

Additional shared resource constraints

- There is a limited number of certified operators to clean machines (see Section 1.3.1).
- The number of IBCs is constrained (see Section 2.3.4).
- The total number of IBCs needed for a job depends on the production route (see Section 2.3.3).
- The number of IBCs needed can vary during an operation, e.g., more IBCs can be needed for filling than for discharging a machine (see Section 2.1.3).
- IBCs are available for production when they are clean (see Section 2.3.4).
- IBCs have transportation times from and to the cleaning station (see Section 1.2).
- The cleaning stations can clean two IBCs at a time (see Section 1.2).
- There can be an initial number of IBCs at the cleaning station at the start of the schedule (see Section 2.3.4).

Further, suitable production routes of jobs depend on the job quantity, product characteristics, and the final packaging unit. Moreover, the processing time of an operation depends on the production route, see Section 2.3.3.

3. Literature review

The main goal of the literature review is to answer the second research question:

"Which methods are available in the literature for our scheduling problem to increase production?"

To answer this research question, Section 3.1 provides an overview of typical scheduling classes to broaden the view on the scheduling research field. Subsequently, we translate our problem into a theoretical problem. This forms the basis to position our scheduling problem in the research field. To achieve this, Section 3.2 reviews and combines several taxonomic frameworks from the literature to classify a broader scope of scheduling problems. In Section 3.3, we use this taxonomy to classify our scheduling problem, as described in Section 2.6. We also classify similar scheduling problems from the literature from the same class according to the taxonomy to position our problem in the research field. Subsequently, we identify gaps and similarities between our scheduling problem and problems in the literature.

After identifying the gaps and similarities, the remainder of the literature review describes modeling techniques for our scheduling problem. Accordingly, Section 3.4 outlines common objective functions and methods to deal with multiple objectives. Section 3.5 identifies and structures solution approaches that might be relevant for our research. This section also reviews the advantages and disadvantages of these methods. Section 3.6 provides a collection of neighborhood structures for our problem. Sections 3.7 and 3.8 describe modeling techniques available in the literature to model sequencing- and resource constraints, respectively. Finally, Section 3.9 summarizes the literature review and provides an overview of suitable models available in the literature that are relevant for our research. We also summarize the gaps in the literature such that we can define our contribution to the scientific body of knowledge.

3.1 Scheduling problems

This section first describes some scheduling terminology. After that, this section provides an overview of typical scheduling classes to broaden the view on the scheduling research field. Broadening the view enhances to identify literature from other scheduling fields that might be relevant for our problem. Subsequently, we translate our problem into a theoretical problem that is well-known in the literature.

Regarding the scheduling terminology, there is a distinction between a schedule and a sequence. A schedule generally corresponds to the allocation of operations to eligible resources over time (Pinto & Grossmann, 1998, p. 433). A sequence usually refers to a job or a permutation of operations on the allocated resources (Pinedo, 2016, p. 23).

To classify scheduling problems, Graham et al. (1979) introduce a triplet notation that consists of the machine environment (α-field), job characteristics and constraints (β), and objective functions (γ). The α-field specifies the scheduling class regarding the machine environment. According to Pinedo (2016) and Ruiz and Vázquez-Rodríguez (2010), typical machine environments are:

Single machine – All jobs need to be processed on a single machine. This case is the simplest of all possible machine environments.

Parallel machines – There are several machines in parallel. Each job has a single operation that needs processing on any one of the machines or a subset thereof. Special versions of this case are *parallel machines with different speeds* and *unrelated parallel machines*. Regarding the former, each machine has its speed, independent of the job. For the latter, the machine speed also depends on the job.

Flow shop – There is a series of multiple machines and each job needs processing on every machine. As the machines are in series, all jobs need processing in the same order, i.e., jobs first go to machine 1, then to machine 2, and so forth.

Flexible flow shop – This class is a combination of the flow shop and the parallel machines classes. There is a series of multiple stages. Each stage has several parallel machines. Every job needs processing at every stage on any one of the parallel machines in that stage. As the stages are in series, all jobs need processing in the same order, i.e., jobs first go to stage 1, then to stage 2, and so forth.

Hybrid flow shop – Similar to the flexible flow shop, this class also has a series of multiple stages with each stage consisting of several parallel machines. A job may skip any number of stages, provided that it needs processing in at least one stage. Within these stages, a job needs processing by any one of the parallel machines that are eligible for that job in that stage.

Job shop – There are multiple machines. Each job needs processing on every machine and each job has its order in which it needs processing on the machines.

Flexible job shop – This class is a combination of the job shop and the parallel machines classes. There are work centers that each consist of parallel machines. Jobs need processing at every work center on any one of the machines within that work center. Moreover, each job has its order in which it needs processing at the work centers.

Open shop – Similar to the job shop, there are multiple machines and each job needs processing on a set of machines. Moreover, the order in which a job needs processing on the machines is unrestricted.

Altogether, our scheduling problem belongs to the hybrid flow shop (HFS) class as our problem consists of multiple stages (e.g., IBC-filling mixing, and packaging), of which some stages consist of multiple parallel machines. Moreover, a job may skip any number of stages, provided that it needs processing in at least one stage (Ruiz & Vázquez-Rodríguez, 2010).

Regarding the job characteristics β-field and the objectives γ-field, Section 3.2 identifies these fields by reviewing several taxonomic frameworks from the literature. Subsequently, we combine these frameworks to a taxonomy to classify a broader scope of scheduling problems.

3.2 Taxonomy of scheduling problems

Section 3.1 outlines a variety of scheduling classes regarding the machine environment (α-field). The job characteristics (β-field) and the objectives (γ-field) also have a wide variety of attributes, i.e., characteristics (e.g., release times or changeovers) (Ribas, Leisten, & Framiñan, 2010). To provide some unification of the diverse attributes, this section reviews several studies that propose taxonomic frameworks for scheduling problems to identify common scheduling problem attributes. We combine these attributes from several studies to create a taxonomic framework that covers a broader scope of scheduling problems.

Reisman et al. (1997) develop an attribute vector description-based taxonomy method. This method classifies, amongst others, vehicle routing problems (Eksioglu, Vural, & Reisman, 2009), data envelopment analyses (Gattoufi, Oral, & Reisman, 2004), and scheduling problems (Cinar, Topcu, & Oliveira, 2015). For the latter, Reisman et al. (1997) review and classify 170 flow shop scheduling studies in the period between 1952 and 1994 based on their type of study (e.g., theoretical or application). Cinar et al. (2015) extend this work by developing a framework to specifically classify studies of flexible job shop problems. They classify 65 studies from the period between 1990 and 2014. Their taxonomy consists of six attribute vectors (e.g., job release time, machine maintenance, and objective functions). Pinto and Grossman (1998) propose a roadmap to classify scheduling problems in manufacturing systems. Their classification method consists of seven attribute vectors (e.g., plant typology and resource constraints). Framiñan et al. (2010) review and classify scheduling problem studies in the period between 1995 and 2010 from a production system point of view.

The studies that propose taxonomic frameworks for scheduling problems each lack attributes; for instance, Pinto and Grossman (1998) do not consider job release times and objective functions, whereas Cinar et al. (2015) do not consider resource constraints. To classify a broader scope of scheduling problems, we combine the classification attributes of Pinto and Grossman (1998), Framiñan et al. (2010), and Cinar et al. (2015).

We use the attribute vector description-based taxonomy method of Reisman et al. (1997) with three branching levels. We group the attribute vectors into the three fields α|β|γ according to the work of Graham et al. (1979). Table 8 provides the resulting taxonomy to classify scheduling problems based on their attributes. For further reference, Appendix 4 elaborates upon the attributes in the taxonomy.

Table 8 | Scheduling problem classification framework

Machine environment (α)		
Machine characteristics	Machine environment	Single machine
		Parallel machines
		Flow shop
		Flexible flow shop
		Hybrid flow shop
		Job shop
		Flexible job shop
		Open shop
	Maintenance	Variable
		Fixed
		None
Job characteristics & constraints (β)		
Job characteristics	Processing time	Operation-dependent
		Stage-dependent
		Machine-dependent
		Fixed
	Release dates	For jobs
		For operations
		None
	Due dates	For jobs
		None
Sequence	Changeovers	Sequence-dependent
		Machine-dependent
		Time-dependent
		Frequency-dependent
		Fixed
		None
	Sequencing constraints	Between jobs
		Between operations
		None
Transport	Transportation times	Variable
		Fixed
		None
	Inventory policy	Unlimited
		Finite
Other characteristics	Demand pattern	Variable
		Fixed (cyclic)
	Time representation	Continuous
		Discrete (fixed slots)
	Resource constraints	Continuous
		Discrete
		None (only machines)
	Lot splitting	Over machines
		None
Objective function (γ)		
Objective	Objective	Makespan
		Flowtime
		Tardiness
		Earliness
		Costs
		Other

3.3 Positioning our research

This section applies the taxonomy in Table 8 to classify our scheduling problem and similar hybrid flow shop (HFS) scheduling problems in the literature to position our problem in the research field. To achieve this, Section 3.3.1 classifies our scheduling problem and Section 3.3.2 classifies similar hybrid flow shop (HFS) scheduling problems in the literature. Subsequently, we identify gaps and similarities between our scheduling problem and problems in the literature.

3.3.1 Classify our scheduling problem

First, we classify our scheduling problem by using the taxonomy framework in Table 8. Regarding the machine environment (α -field), our scheduling problem belongs to the HFS class as Section 3.1 identifies. Furthermore, the maintenance policy is variable since the starting times of maintenance activities are flexible as Section 2.2.2 discusses.

Regarding job characteristics and constraints (β-field) of our scheduling problem, the processing times depend on the product quantity, the stage, and the machine, as Section 2.3.2 discusses. Both the jobs and machines may have release times. When a machine has a release time, this can be modeled by allocating release times to all the operations that need processing on these machines. Moreover, only jobs may have due dates as Section 2.2.2 explains.

Changeovers depend on the job sequence and the machine as Section 2.2.2 describes. Additionally, there are sequencing constraints between the operations of a job (e.g., first mixing, then packaging). Also, jobs have sequencing constraints since certain job sequences are not allowed due to restrictions on the product claims as Section 2.2.2 describes.

The transportation times between the stages are variable since these depend on the machine allocation. Besides that, intermediate storage capacity is finite since a limited number of IBCs store the products between the stages. Products that are stored in big-bags can be stored in the high-rise warehouse, which has sufficient capacity.

The demand pattern is variable since most jobs are order-based as there is little similarity between demand patterns in different scheduling periods as Section 2.2.1 describes. Besides that, the plant is producing continuously and there are no time slots. Nevertheless, this does not restrict modeling with time slots. Moreover, operations of different jobs may require additional resources (e.g., operators or IBCs) in different stages simultaneously. These resources have a finite capacity. Moreover, the required resources may vary while processing an operation, e.g., more IBCs can be needed to fill a mixer than to discharge a mixer as Section 2.1.3 describes. Besides that, lot splitting is not possible.

Finally, the objective function (γ-field) of the scheduling problem is twofold: (i) minimizing the number of cleanings and (ii) minimizing the tardiness of jobs.

3.3.2 Classify and compare related scheduling problems

Next, we classify similar HFS scheduling problems in the literature such that we can position our problem in the research field. Subsequently, we identify gaps and similarities between our scheduling problem and HFS problems in the literature.

There are several reviews of HFS problems in the literature, like those of, Linn and Zhang (1999), Wang (2005), Quadt and Kuhn (2007), Ribas et al. (2010), or more recently, Cinar et al. (2015) and Li et al. (2020). These studies thoroughly review HFS problems in the literature over the last 20 years. Therefore, we use their work as a starting point to identify HFS problems in the literature that are similar to our problem. However, most studies that the aforementioned authors review are theoretical with synthetic instances that are not verified in practice (Cinar, Topcu, & Oliveira, 2015, pp. 22-23). Therefore, we extend our search to similar scheduling problems by using the backward- and forward snowballing technique, i.e., using the references of papers to identify additional papers. Table 9 provides an overview of the classification of our scheduling problem as discussed in Section 3.3.1. Moreover, Table 9 classifies 10 scheduling problems from the literature that, to the best of our knowledge, are most similar to our problem.

We note from Table 9 that almost every characteristic of our problem appears in at least one study in the literature of HFS. We note that the HFS problems in the literature each cover only a small variety of practical constraints (e.g., release times, transportation times, or resource capacity constraints), which Cinar et al. (2015, p. 34), and Li et al. (2020, p. 73) also experience, since most studies do not focus on practical problems. Moreover, there appears to be no scheduling problem with a similar set of characteristics as our problem.

Furthermore, most of the problems that Table 9 classifies have the objective to minimize the makespan, tardiness, or both. Cinar et al. (2015, p. 33) and Li et al. (2020, p. 73) highlight that the makespan is the most common objective. Currently, Euroma minimizes the number of cleanings and the tardiness of jobs, as Section 2.2.2 describes. Nevertheless, we question whether these objectives are most suitable to increase the production throughput, which is the goal of Euroma, as Section 1.3.1 describes. For instance, minimizing the makespan may be relevant as it optimizes the use of limited resources (Ruiz & Vázquez-Rodríguez, 2010, p. 22). Moreover, minimizing the makespan increases machine utilization and throughput (Minella & Ruiz, 2008), which corresponds with the goal of Euroma.

After identifying the gaps and similarities, the remainder of the literature review describes modeling techniques that are suitable for our scheduling problem. Each section elaborates upon the techniques and methods that HFS studies from the literature apply. Nevertheless, we broaden the scope by reviewing other scheduling fields and even other research fields if applicable.

The remainder of this chapter is organized as follows. We question whether the current objective of Euroma is most suitable to increase the production throughput, which is the main goal of Euroma. Therefore, Section 3.4 provides a more in-depth review of objective functions that might be suitable to improve, amongst others, the production throughput. Besides, this section outlines several methods to deal with multiple objectives. Section 3.5 provides an overview of solution approaches and elaborates upon the approaches that are relevant to our research. Accordingly, Section 3.6 explains several neighborhood structures and touches upon their connectedness. Sections 3.7 and 3.8 describe modeling techniques available in the literature to model sequencing- and resource constraints, respectively. Finally, we summarize the literature review in Section 3.9.

Table 9 | Problem classification

			Our problem	(Ebrahimi, et al., 2014)	(Ruiz, et al., 2008)	(Tao, et al., 2020)	(Karimi, et al., 2017)	(Naderi, et al., 2009)	(Gao, Gen, et al., 2006)	(Jun & Park, 2015)	(Botta, et al., 2000)	(Costa, et al., 2020)	(Voß & Witt, 2007)
Machine environment (α)													
Machine characteristics	Machine environment	Single machine											
		Parallel machines											
		Flow shop											
		Flexible flow shop											
		Hybrid flow shop	X	X	X	X	X	X	X	X	X	X	X
		Job shop											
		Flexible job shop											
		Open shop											
	Maintenance	Variable	X						X				X
		Fixed											
		None		X	X	X	X	X		X	X	X	
Job characteristics & constraints (β)													
Job characteristics	Processing time	Operation-dependent	X	X	X	X	X	X	X	X	X		X
		Stage-dependent	X	X	X	X						X	
		Machine-dependent	X		X		X		X	X	X		
		Fixed											
	Release dates	For jobs	X										
		For operations	X		X								
		None		X		X	X	X	X	X	X	X	X
	Due dates	For jobs	X	X				X		X	X		X
		None			X	X	X		X			X	
Sequence	Changeovers	Sequence-dependent	X	X	X			X					X
		Machine-dependent	X		X								
		Time-dependent											
		Frequency-dependent											
		Fixed									X	X	
		None				X	X		X	X			
	Sequencing constraints	Between jobs	X		X								
		Between operations	X		X	X					X		
		None		X			X	X	X	X		X	X
Transport	Transportation times	Variable	X		X		X	X			X		X
		Fixed											
		None		X		X			X	X		X	
	Inventory policy	Finite	X							X		X	X
		Unlimited		X	X	X	X	X	X		X		
Other characteristics	Demand pattern	Variable	X	X	X	X	X	X	X	X	X	X	X
		Fixed (cyclic)											
	Time representation	Continuous	X	X	X	X	X	X	X	X	X	X	X
		Discrete (fixed slots)											
	Resource constraints	Continuous	X										
		Discrete				X						X	
		None (only machines)		X	X		X	X	X	X	X		X
	Lot splitting	Over machines											
		None	X	X	X	X	X	X	X	X	X	X	X
Objective function (γ)													
Objective	Objective	Makespan		X	X	X	X	X	X			X	
		Flowtime											
		Tardiness	X	X				X		X	X		X
		Earliness											
		Costs											
		Other	X			X			X				

3.4 Objective functions

We question whether the current objective of Euroma (i.e., minimizing the number of cleanings and the tardiness) is most suitable to improve, amongst others, the production throughput, which is the main goal of Euroma. Therefore, this section provides a more in-depth review of objective functions that might be suitable for our problem. Besides, as our problem has multiple objectives, this section elaborates upon several methods to deal with multiple objectives.

At first, the following objectives are suitable for our problem. Minimizing the makespan increases machine utilization and throughput (Yenisey & Yagmahan, 2014), which is in line with the main goal of Euroma. Moreover, earliness and tardiness objectives provide a customer-centric approach (Tahmasbi & Moghaddam, 2011), which corresponds with the current objective of Euroma. Besides that, minimizing the flowtime enhances a stable usage of resources and less work-in-progress (Yenisey & Yagmahan, 2014). Minimizing the flowtime might be promising for our problem to create a stable flow of IBCs through the process. Nevertheless, note that a combination of the aforementioned methods might conflict.

The majority of the literature concentrates on single objectives. However, Yenisey and Yagmahan (2014, p. 119) and Li et al. (2020, p. 73) address that single-objective criteria are insufficient for practical applications as these are multi-objective by nature. There are several ways to formulate an objective function f. Table 10 provides different formulations for this function, similar to the work of T'Kindt and Billaut (2006), and Yenisey and Yagmahan (2014).

Table 10 | Objective function formulation

Formulations	
Z	Single objective; minimize Z
$f_w(Z_1, Z_2, \dots, Z_k)$	Minimize weighted k objectives (utility approach)
$f_p(Z_1, Z_2, \dots, Z_k)$	Minimize all objectives (Pareto-optimal approach)
$f_{np}(Z_1, Z_2, \dots, Z_k)$	Minimize all objectives, each objective is evaluated separately
$f_H(Z_1, Z_2, \dots, Z_k)$	Minimize the objectives in hierarchical order (hierarchical approach)
$f_\varepsilon(Z_p \mid Z_1, Z_2, \dots, Z_k)$	Minimize Z_p, and the other k objectives are subject to constraints (ε–constraint)
$f_{gp}(Z_1, Z_2, \dots, Z_k)$	Minimize each objective until their individual goal is reached (goal-seeking)

There are three approaches to deal with multi-objective functions (T'Kindt & Billaut, 2006; Minella & Ruiz, 2008; Yenisey & Yagmahan, 2014). These three approaches, including examples of how to deal with multi-objective formulations such as in Table 12 are:

1. In the *a priori approach*, a decision-maker needs to provide input to the objective function before generating a solution. This approach is possible for, amongst others, $f_w(Z_1, Z_2, \dots, Z_k)$ and $f_H(Z_1, Z_2, \dots, Z_k)$. Regarding the former, the decision-maker determines the weights of the single objectives a priori, e.g., $f_w = wZ_1 + (1 - w)Z_2$, where $0 \leq w \leq 1$. For the latter, the decision-maker determines the hierarchical order of the objectives a priori. For instance, for $f_H(Z_1, Z_2, \dots, Z_k)$, find the optimal solution of Z_1 first, then optimize Z_2 subject to Z_1, and so on. This process continues until Z_k is optimal subject to Z_{k-1}.
2. In the *a posteriori approach*, a set of Pareto-optimal solutions is developed. The decision-maker selects a solution from this set, i.e., $f_p(Z_1, Z_2, \dots, Z_k)$.
3. During the *interactive approach*, the decision-maker interactively provides the preferences of the weights of the objectives during the solution process. The weights are iteratively calibrated until there is a suitable compromise between the objectives.

3.5 Solution approaches

In this section, the aim is to identify suitable solution approaches for our scheduling problem. To achieve this, this section first categorizes and elaborates upon the different solution approaches available in the literature regarding HFS problems. After that, the remainder of this section concentrates on promising solution approaches for our problem.

Regarding solution approaches for HFS problems, we concentrate on studies from 1995 or later due to the significant development of new approaches starting from 2000 (Cinar, Topcu, & Oliveira, 2015). A review of solution approaches before 1995 is available in the work of Linn and Zhang (1999). Ribas et al. (2010) provide a comprehensive review of proposed solution approaches for HFS problems. Moreover, Ruiz and Vázquez-Rodríguez (Ruiz & Vázquez-Rodríguez, 2010) present a literature review of exact, heuristic, and metaheuristic methods that have been proposed for HFS problems. Figure 22 provides a classification of solution approaches for HFS problems based on the classifications of Ribas et al. (2010) and Ruiz and Vázquez-Rodríguez (2010).

Since HFS problems are generally strongly NP-hard, exact approaches can generally only solve instances up to 15 to 20 jobs with 5 stages to optimality (Ribas, Leisten, & Framiñan, 2010). Despite the relative success, exact approaches are still incapable to solve medium and large problem instances for real-world problems (Ruiz & Vázquez-Rodríguez, 2010).

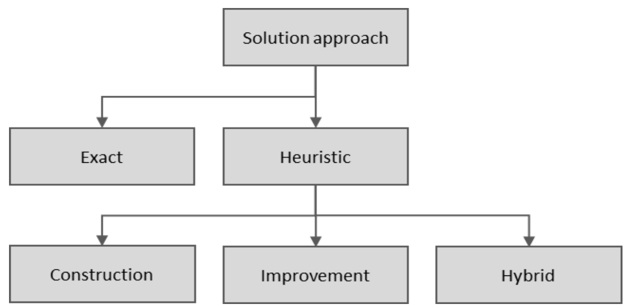


Figure 22 | Solution approach classification

Nevertheless, there is a variety of non-exact and efficient heuristics that can obtain good solutions for large problem instances in reasonable computational time. These heuristics can be partitioned into construction-, improvement-, and hybrid heuristics. Construction heuristics construct solutions from scratch and improvement heuristics improve these solutions. Hybrid approaches combine several modeling methods. For instance, a hybrid approach can use an improvement heuristic to allocate jobs to machines and an exact approach to find the best sequence on each machine (Ribas, Leisten, & Framiñan, 2010).

Our problem consists of instances of up to 400 jobs, as described in Section 1.2. Therefore, exact approaches seem to be not suitable. Therefore, the remainder of this section concentrates on promising heuristics for our problem. Accordingly, Section 3.5.1 describes several construction heuristics and Section 3.5.2 elaborates upon improvement heuristics.

3.5.1 Construction heuristics

There are several construction heuristics for HFS problems in the literature, mostly concerning the 2-stage HFS problem (Ribas, Leisten, & Framiñan, 2010). This section reviews and describes some construction heuristics that are suitable for k-stage HFS problems.

At first, Nawaz et al. (1983) propose a construction heuristic (NEH) that is commonly used in the literature (Ruiz, Sivrikaya Şerifoğlu, & Urlings, 2008). In essence, the NEH is applicable to scheduling problems where every job $j \in J$ can process on any one of the eligible machines of the subset $M_j \subseteq M$ with a processing time of $p_{j,k}$, where $k \in M_{j,i}$.

The NEH consists of the following steps. First, list all jobs in a descending order based on their average processing time, $APT_j = \frac{\sum_{k \in M_j} p_{j,k}}{|M_j|}$. Second, take the first job of the list with the highest APT_j and define every possible schedule. Select the schedule with the best objective value. Next, insert the third job of the list on the position into the schedule that results in the best objective value, without changing the relative positions of the already scheduled jobs. This process continues until all jobs are in the schedule.

Ruiz et al. (2008) compare several construction heuristics, amongst others, the NEH heuristic. They modify the NEH heuristic such that it can deal with sequencing relations between the operations of a job, i.e., operations cannot process before any of its predecessors and no later than any of its successors. To achieve this, Ruiz et al. (2008) consider a problem where every job $j \in J$ consists of consecutive operations $O_j = \{1,2,..,n_j\}$, where the i^{th} operation of job j is denoted by $o_{j,i}$. An operation $o_{j,i}$ can process on any one of the eligible machines of the subset $M_{j,i} \subseteq M$. Each operation $o_{j,i}$ requires a processing time of $p_{j,i,k}$ on machine $\mathrm{k} \in M_{j,i}$. They extend the list sorting rule such that the total average processing time is $TAPT_j = \sum_{i \in O_j} \frac{\sum_{k \in M_{j,i}} p_{j,i,k}}{|M_{j,i}|}$. When inserting the next job of the list into the schedule, they consider the positions in the schedule that do not violate the sequencing relations. Ruiz et al. (2008) found that the NEH heuristic is vastly superior to the other construction heuristics that they consider for HFS problems. Liu, Yan, and Price (2017) extend the NEH heuristic by adding a tie-breaking rule. They show that their extension can find slightly better solutions for some problem instances.

Moreover, Guinet and Solomon (1996) propose a two-phase construction heuristic to schedule jobs to minimize the tardiness or makespan. They first sort the jobs based on priority rules, i.e., a measure that defines the sorting (e.g., APT_j). Then, they schedule the jobs based on this order. Their results show that the NEH heuristic outperforms all other approaches. Besides that, Ruiz and Marato (2006) propose an effective heuristic that considers changeover times and release dates for machines. Their heuristic assigns an operation to the machine that can finish it at the earliest possible time.

3.5.2 Improvement heuristics

There is a large variety of improvement heuristics available. In essence, improvement heuristics start with an initial solution S. Neighborhood operators (see Section 3.6) try to improve S by making changes to it (e.g., swapping two jobs in the sequence) to get a neighbor solution S'. Improvement heuristics each handle and accept neighbor solutions differently. Nevertheless, each improvement heuristic returns at the end the best solution found S^*. This section reviews a variety of improvement heuristics that have been applied in the HFS field. Moreover, this section elaborates upon promising improvement heuristics for our problem to identify their strengths and weaknesses.

For HFS problems, commonly applied improvement heuristics are simulated annealing (SA), tabu search (TS), and genetic algorithms (GA). Less frequently used heuristics are artificial immunes (AIS), neural networks (NN), and ant colony optimization (ACO) (Ruiz & Vázquez-Rodríguez, 2010; Cinar et al., 2015).

Kirkpatrick (1983) introduces simulated annealing (SA). This method always accepts the neighbor solution S' when it is better than the current solution S. When the neighbor solution S' is worse than the current solution S, SA includes randomness in the acceptance criterion to be able to escape local optima (Ruiz & Vázquez-Rodríguez, 2010). In this case, the neighbor solution S' is accepted with a probability that depends on the difference between the objective values $F(S)$ and $F(S')$, and the progression of the heuristic, which is often denoted by the temperature T. The latter decreases over time with a factor α, which is referred to as the cooling factor. Therefore, the acceptance probability decreases over time such that it is less likely that SA accepts worse solutions over time. The heuristic stops when $T < T_{stop}$. Finally, SA returns the best solution found S^*. Figure 23 provides pseudo code of SA with a minimization objective. Low (2005) applies SA to minimize the total flow time of an HFS problem with sequence-dependent changeover times. Naderi et al. (2009) also use SA to solve an HFS problem with sequence-dependent changeovers. In similar work, SA is the improvement heuristic to solve an HFS problem with both sequence-dependent changeovers and transportation times between stages (Naderi, Zandieh, Khaleghei, & Roshanaei, 2009). Jin et al. (2006) propose two SA heuristics that differ in their neighborhood structures. They conduct an extensive computational study and show the efficiency of these heuristics. They compare their SA heuristic with the tabu search (TS) heuristic of Riane et al. (2002) and show similar results.

Simulated annealing

Construct the initial solution S
Initialize: $S^* \leftarrow S, T \leftarrow T_{start}, T_{stop}, Len, \alpha$
While $T > T_{stop}$ **do**
    For $k = 1$ **to** Len **do**
        $S' \leftarrow$ GenerateNeighbor(S)
        If $F(S') < F(S^*)$ **then**
            $S^* \leftarrow S'$
        End
        If $F(S') \leq F(S)$ **or** $\exp\left(-\frac{F(S)-F(S')}{T}\right) < \text{random}(0,1)$ **then**
            $S \leftarrow S'$
        End
    End
    $T \leftarrow \alpha T$
End
Return S^*

Figure 23 | Simulated annealing pseudo-code

Genetic Algorithms (GA) are inspired by the natural selection process. There are many variations of GAs. In essence, GAs use biologically inspired operators (e.g., crossover, selection, and mutation) to generate a population of neighbor solutions with a better fitness value. As there are many variations of GAs, we do not provide pseudo code, instead, we refer to the work of Gen and Cheng (1999). Regarding the application of GA, Ruiz and Marato (2006) propose a GA for an HFS problem with sequence-dependent changeover times to minimize the makespan. Their GA outperforms several other GA heuristics available in the literature. Moreover, their GA obtains schedules that are better than the ones generated manually by the personnel of a real-world production shop. Yaurima et al. (2009) propose a similar GA with the extension to deal with limited buffers between stages.

The improvement heuristic TS has a tabu memory list that stores the last neighbor solutions visited. The idea of this tabu list is to avoid going back and forth between neighbors. In essence, TS generates a set of neighbor solutions that are not in the tabu memory list. The neighbor S' with the best objective value gets accepted. This way, TS might accept a neighbor S' with a worse objective value $F(S')$ than the objective value of the current solution $F(S)$ to avoid getting stuck in local optima. In the case that the neighbor S' is in the tabu list, an aspiration criterion needs to be satisfied to override the tabu state of that neighbor. The heuristic stops based on a stopping criterion, e.g., maximum computational time reached (Glover & Laguna, 1998). Figure 24 provides a pseudo-code of TS with a minimization objective.

```
Tabu search
Construct the initial solution S
Initialize: S* = S, maxTabuCount
While not stopping condition do
    neighborList ← getNeighbors(S)
    For S' in neighborList do
        If not tabuList.hold(S') and F(S') < F(S) then
            S ← S'
        Elseif aspirationCriterion(S')
            S ← S'
        End
    End
    If F(S) < F(S*) then
        S* ← S'
    End
    tabuList.add(S)
    If tabuList.count > maxTabuCount then
        tabuList.removeLast
    End
End
Return S*
```

Figure 24 | Tabu search pseudo-code

Regarding the application of TS, Jin et al. (2006) use TS for circuit printing production scheduling and Chen et al. (2007) apply TS for a container handling system problem. Wang and Tang (2009) propose a TS application to minimize the weighted completion times objective subject to a finite buffer capacity between stages.

Some studies apply less frequently used heuristics, e.g., Alisantoso et al. (2003) and Engin and Doyen (2004) apply an artificial immunes system (AIS) heuristic to solve HFS problems. Moreover, Wang et al. (2003) and Tang et al. (2005) apply Neural Networks (NN) to minimize the makespan. Tang et al. (2005) extend the NN method by considering sequence-dependent changeover times. However, the NN approaches are complex and seem to result in relatively poor solutions compared to other heuristics in the field (Ruiz & Vázquez-Rodríguez, 2010). Ying and Lin (2009) propose an ant colony optimization (ACO) heuristic and outperform Janiak et al. (2004). Tseng et al. (2008) propose a particle swarm optimization (PSO) heuristic which in turn was superior to the PSO that Ying and Lin (2009) propose.

In this section, we touch upon many heuristics that different studies propose. Comparing these heuristics based on their performance would be interesting. However, due to the diversity of the HFS problems and their instances (e.g., Table 9), a literature-based comparison seems to be almost impossible. Despite a few papers that compare different heuristics in similar settings, an overall superior approach is hard to identify.

3.6 Neighborhood structures

This section elaborates upon neighborhood structures available in the literature of HFS problems. A neighborhood search technique (also referred to as a neighborhood operator) mutates a solution to find a better solution according to the objective function. The set of neighborhood operators (i.e., the neighborhood structure) defines the size of the neighborhood. Moreover, a neighborhood is connected when the neighborhood structure can transform any solution into any other solution in a finite set of iterations. Concerning optimality, a neighborhood is opt-connected if any initial solution can transform into an optimal solution in a finite set of iterations (Kupfahl & Bierwirth, 2016).

At first, Table 11 describes common neighborhood operators in the HFS field. Moreover, Table 12 provides a small selection of papers that consider these operators. The remainder of this section reviews different configurations of neighborhood structures regarding their size and connectedness.

Table 11 | Neighborhood operators

Nr	Neighborhood operator
1	Swap two operations on the same machine
2	Swap two operations between alternative machines
3	Move one operation to another position on the same machine
4	Move one operation to a position on an alternative machine
5	Move one operation to the same position on an alternative machine
6	Move the operations between two positions to another position on the same machine
7	Move the operations between two positions to an alternative machine
8	Inverse the sequence of the operations between two positions on the same machine

Table 12 | Neighborhood operators in HFS problems

	Neighborhood operator Nr							
Papers	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
(Al-harkan & Qamhan, 2019)	X	X	X	X				
(Karimi, Ardalan, Naderi, & Mohammadi, 2017)	X				X	X	X	
(Kupfahl & Bierwirth, 2016)	X	X						X
(Naderi, Zandieh, Khaleghei, & Roshanaei, 2009)		X						X
(Naderi, Zandieh, & Roshanaei, 2009)	X	X	X	X				X
(Wang & Tang, 2009)	X	X						X
(Zhang, Zhang, Song, Wang, & Zhou, 2019)	X		X	X				

The operators in Table 11 require selecting operations, machines, positions in a sequence, or a combination of these. There are several strategies to make these selections to focus on promising neighbors. For instance, Jun and Park (2015) use a rule to allocate a machine to a job to minimize the total tardiness. This allocation rule considers the processing time of the job on the alternative machines and assigns the job to the machine that will complete the process soonest. Nevertheless, most papers make these selections randomly, e.g., the work of Naderi et al. (2009), Karimi et al. (2017), or Al-harkan and Qamhan (2019).

Furthermore, there are several strategies to select neighborhood operators. For instance, Naderi, Zandieh, and Roshanaei (2009) make a tradeoff between small neighbors, e.g., operators (1) and (2), to intensify the search space and conversely large neighborhoods, e.g., operator (8) or three consecutive operations of (1) or (2), to diversify the search space. They first intensively search the current search space by using the small neighborhood operators. After having searched the current space, they use large neighborhood operators to identify new search spaces. Wang and Tang (2009) use a TS heuristic with the operators (1) and (2). When the TS heuristic rejects the solutions provided by these operators, an inversion operator (8) is used. Moreover, Dauzère-Pérès (2018) conduct preliminary neighbor evaluations to identify promising neighbors. These evaluations are based on the calculation of the lower bounds on the new makespan. When there are no improvements for a predefined number of iterations, their neighborhood diversifies by swapping operations arbitrarily. Nevertheless, many studies select neighborhood operators randomly, e.g., the work of Karimi et al. (2017) or Zhang et al. (2019).

Besides that, neighborhood operators can result in infeasible schedules (Dauzère-Pérès, Shen, & Neufeld, 2018). For example, in HFS problems, jobs can only move to eligible machines (Zhang, Zhang, Song, Wang, & Zhou, 2019). Moreover, in more complex HFS problems with sequencing constraints between jobs, some sequences are not allowed. Therefore, verifying the feasibility of the neighborhood operation is important.

3.7 Sequencing constraints

In HFS problems, there are sequencing constraints (also referred to as precedence constraints) between the consecutive operations of a job (Ruiz & Vázquez-Rodríguez, 2010). For instance, regarding our problem, mixing is required before packaging. Section 3.7.1 reviews techniques to model these sequencing constraints. This section also describes techniques to extend these sequencing constraints to model transportation times between operations and sequence-dependent changeovers between two consecutive jobs on the same machine. Moreover, this section provides a technique to model release times of operations. Besides that, Section 3.7.2 reviews several techniques to model sequencing constraints between different jobs.

3.7.1 Sequencing relations between operations of a job

Recall that in HFS problems every job $j \in J$ consists of a sequence of consecutive operations $O_j = \{1,2,..,n_j\}$, where the i^{th} operation of job j is denoted by $o_{j,i}$. An operation $o_{j,i}$ can process on any one of the eligible machines of the subset $M_{j,i} \subseteq M$. Each operation $o_{j,i}$ requires a processing time of $p_{j,i,k}$ on machine $k \in M_{j,i}$. The start- and finish times of an operation $o_{j,i}$ are denoted by $s_{j,i}$ and $f_{j,i} \geq s_{j,i} + p_{j,i,k}$, respectively. Note that when $f_{j,i} > s_{j,i} + p_{j,i,k}$, preemption on machine k is allowed. There are common sequencing constraints in HFS problems (Ruiz & Vázquez-Rodríguez, 2010). These constraints are also applicable to our problem. At first, the operations O_j of a job j should not overlap, thus, $o_{j,i}$ cannot start earlier than $o_{j,i-1}$ is finished, i.e.,

$$s_{j,i} \geq f_{j,i-1} \qquad \forall j \in J,\ i \in O_j,\ \text{s.t. } i > 1. \tag{1a}$$

Note that constraint (1a) assumes that consecutive operations O_j of a job j have no time legs in between them (e.g., transportation times). Karimi et al. (2017) extend constraint (1a) to model time lags between consecutive operations O_j of a job j. To achieve this, denote the time lag between a pair of consecutive operations $(o_{j,i-1}; o_{j,i})$ of a job j by $lag_{j,i-1,i}$ and modify constraint (1a) such that

$$s_{j,i} \geq f_{j,i-1} + lag_{j,i-1,i} \qquad \forall j \in J,\ i \in O_j,\ s.t.\ i > 1. \tag{1b}$$

Besides that, a machine k can only process one job at a time. To achieve this, let $(o_{j',i'}; o_{j,i}) \in \Pi_k$ be a set of consecutive operation pairs in the sequence of machine k, then

$$s_{j,i} \geq f_{j',i'} \qquad \forall (o_{j',i'}; o_{j,i}) \in \Pi_k,\ k \in M_{j,i} \cap M_{j',i'},\ s.t.\ o_{j,i} \neq o_{j',i'}. \tag{2a}$$

Constraint (2a) assumes that the consecutive operations in the sequence of machine k have no time legs in between them (e.g., setup times). Nevertheless, Dauzère-Pérès et al. (2018) extend constraint (2a) to include setup times. To achieve this, denote the setup time between a pair of consecutive operations on the same machine $(o_{j',i'}; o_{j,i}) \in \Pi_k$ by $setup_{(j',i'),(j,i),(k)}$ and modify constraint (2a) such that

$$s_{j,i} \geq f_{j',i'} + setup_{(j',i'),(j,i),(k)} \quad \forall (o_{j',i'}; o_{j,i}) \in \Pi_k,\ k \in M_{j,i} \cap M_{j',i'},\ s.t.\ o_{j,i} \neq o_{j',i'}. \tag{2b}$$

Furthermore, the operation o_{ji} of a job j may require a release time, which is denoted by r_{ji}. In this case, an operation o_{ji} cannot start earlier than r_{ji}. Ruiz et al. (2008) model this as follows

$$s_{j,i} \geq r_{j,i} \qquad \forall j \in J,\ i \in O_j. \tag{3}$$

Regarding our problem, the constraints (1b), (2b), and (3) are applicable as our problem has, respectively, transportation times between two consecutive operations of the same job, changeover times between two consecutive operations on the same machine, and operations of jobs mays have release times. To achieve this, the starting time s_{ji} of every operation o_{ji} should satisfy the following three constraints

$$s_{j,i} \geq f_{j,i-1} + lag_{j,i-1,i} \qquad \forall j \in J,\ i \in O_j,\ s.t.\ i > 1 \tag{1b}$$
$$s_{j,i} \geq f_{j',i'} + setup_{(j',i'),(j,i),(k)} \quad \forall (o_{j',i'}; o_{j,i}) \in \Pi_k,\ k \in M_{j,i} \cap M_{j',i'},\ s.t.\ o_{j,i} \neq o_{j',i'} \tag{2b}$$
$$s_{j,i} \geq r_{j,i} \qquad \forall j \in J,\ i \in O_j. \tag{3}$$

3.7.2 Sequencing relations between jobs

In scheduling problems, there may be sequencing constraints between jobs. For instance, regarding our problem, a solution is infeasible when sequencing a halal-certified job after a haram job as described in Section 2.2.2. However, despite the interest in scheduling problems over the past years, the literature on sequencing constraints between jobs in scheduling problems is limited (Afzalirad & Rezaeian, 2017).

Nevertheless, according to Sun et al. (2010), there are three options to attain feasibility when considering sequencing constraints: (i) reject an infeasible solution without consideration, (ii) repair the infeasible solutions such that they become feasible, or (iii) allow infeasibility and add a penalty to worsen the objective function. Regarding the first option, Driessel and Mönch (2011) consider a scheduling problem with an arbitrary set C of paired jobs with sequencing constraints, i.e., job j' must precede job j if $(j', j) \in C$. For each job, Driessel and Mönch determine feasible positions in the schedule and do not consider the infeasible solution space. The drawback of this method is that it can be very time-consuming to find a good solution, especially when the search space contains many infeasible solutions (Sun, Cheng, & Liang, 2010).

Besides that, Afzalirad and Rezaeian (2017) consider the same sequencing constraints as (Driessel & Mönch, 2011). However, they allow infeasibility and use a corrective algorithm to attain feasible solutions. Moreover, Sun et al. (2010) expand their search space to the infeasible regions. Once a solution violates a constraint, a penalty worsens the objective. This penalty can increase as the algorithm progresses such that it is more likely to end up with a feasible solution.

3.8 Additional shared resource constraints

Regarding our problem, an operation may require several IBCs to fill and discharge machines, and to store products between stages (e.g., mixing and packaging). Moreover, a cleaning may require several certified operators. These resources are limited and may be required by different stages (e.g., IBC-filling, mixing, and packaging) simultaneously, as described in Sections 1.3.1 and 2.3.4. Costa et al. (2020) refer to limited resources that may be required in different stages simultaneously as additional resources. Concerning our problem, the IBCs and the certified operators are additional resources.

According to Blazewicz et al. (2007), additional resources can be renewable, non-renewable, or both. Renewable resources are limited and are reusable, e.g., AGVs, operators, or equipment. Non-renewable resources are consumed during operations, e.g., raw materials or energy. Moreover, resources can be considered discrete when they are consumed at a constant level during the process, or continuous, in which the resource consumption differs during the process (Pinto & Grossmann, 1998).

Regarding our problem, the number of certified operators required during cleaning is constant, as described in Section 1.3.1. However, the number of IBC required while processing an operation on a machine may be variable. For instance, there may be more IBCs required to fill a mixer than to discharge a mixer, as described in Section 2.1.3. In this case, IBCs that are not required to discharge a mixer go to the IBC-cleaning station immediately after they have filled the mixer, while the other IBCs wait to discharge the product from the mixer. For these reasons, we are interested in techniques to model limited renewable additional resources with constant- and continuous consumption, since these are present in our problem.

As additional resources are limited, it is necessary to consider feasible combinations of simultaneously processed operations that use the same limited resources at each point in time (Edis, Oguz, & Ozkarahan, 2013). To achieve this, the overall resource capacity needs to be sufficient to ensure feasibility (Brucker & Krämer, 1996).

Costa et al. (2020) state that their work is the first study in which additional renewable resources in an HFS scheduling problem are considered. They consider a critical workforce capacity (i.e., at each stage, the number of operators is lower than the number of machines) for performing changeovers. Moreover, they assume that exactly one operator is required during every changeover process. They propose a discrete backtracking search algorithm to solve this problem. In essence, this algorithm calculates at every point in time how many operators are required. If the critical workforce capacity is exceeded, a changeover postpones to the first point in time an operator is available. Therefore, feasibility regarding the resource constraint is assured (Costa, Fernandez-Viagas, & Framiñan, 2020).

Shortly after the work of Costa et al. (2020), Tao et al. (2020) consider an HFS problem with different resources (e.g., tools). A job may require multiple different resources depending on the machine. These resources may be required by several jobs in different stages simultaneously. Moreover, the resource usage is constant while processing a job. Tao et al. (2020) model these resource constraints similar to Costa et al. (2020); if the required resources are not available, the job postpones until the required resources are available.

To the best of our knowledge, the work of Costa et al. (2020) and Tao et al. (2020) are the only studies that consider resource constraints in an HFS environment. Nevertheless, there are some studies in other scheduling domains that consider resource constraints. Blazewicz et al. (2007) review the general area of resource-constrained scheduling problems. Edis et al. (2013) extend this work with a more in-depth review of studies on parallel machine scheduling problems with resource constraints. Both Edis et al. (2013) and

Costa et al. (2020) find gaps in the literature regarding resource constraints in scheduling. They state that most studies focus on a single resource type, thus, versions with multiple resource types are potential research areas. Moreover, there are almost no studies where the same additional resources can be used in multiple different stages, as most studies only focus on shared resources within one stage (Edis, Oguz, & Ozkarahan, 2013). Furthermore, future research may focus on resource constraints in HFS scheduling problems (Costa, Fernandez-Viagas, & Framiñan, 2020).

3.9 Summary of the literature review

The main goal of the literature review is to answer the second research question: Which methods are available in the literature to solve our scheduling problem to increase the production throughput? To answer this question, Section 3.1 outlines several scheduling classes to broaden the view on the scheduling research field. Accordingly, we identify our scheduling problem as a hybrid flow shop (HFS) problem.

In Section 3.2, we combine several taxonomic frameworks in the literature to cover a broader scope of scheduling problems. Subsequently, Section 3.3 classifies our problem and similar HFS problems in the literature by using the combined taxonomy (see Table 9) such that we can position our problem in the research field. Consequently, we identify from the classification that almost every characteristic of our problem appears in at least one study in the literature of HFS. However, HFS problems in the literature each cover only a small variety of practical characteristics (e.g., release times, transportation times, or resource capacity constraints), which Cinar et al. (2015, p. 34), and Li et al. (2020, p. 73) also experience. Moreover, there appears to be no scheduling problem with a similar set of characteristics as our problem. Section 3.4 identifies that the majority of the literature concentrates on single objectives. However, single-objectives are insufficient for practical applications as Minella and Ruiz (2008), Lei (2009), and Yenisey and Yagmahan (2014) address. Moreover, there is a gap in the literature regarding suitable objectives for HFS scheduling problems with practical characteristics (e.g., sequence-dependent changeovers or resource constraints) (Li, Gao, & Peng, 2020, p. 73). As our problem has multiple objectives, Section 3.4 identifies methods to formulate and solve multi-objective functions. Finally, promising objectives for our problem are minimizing the makespan and flowtime. The former increases machine utilization and the latter enhances a stable usage of resources and less work-in-progress, which both correspond with the objectives of our scheduling problem.

Section 3.5 highlights that HFS problems are generally strong NP-hard (Ribas, Leisten, & Framiñan, 2010). Therefore, exact approaches are incapable to solve medium and large problem instances for real-world problems (Ruiz & Vázquez-Rodríguez, 2010). Nevertheless, this section describes a variety of non-exact and efficient heuristics that can obtain good solutions for large problem instances in a reasonable time. Section 3.5.1 elaborates upon a promising construction heuristic of Nawaz et al. (1983) and Section 3.5.2 describes several promising improvement heuristics that studies propose. There is a large diversity of HFS problems and their instances. Therefore, a literature-based comparison seems to be almost impossible to identify an overall superior approach. Nevertheless, commonly applied and promising heuristics for HFS problems seem to be simulated annealing, tabu search, and genetic algorithms.

Section 3.6 outlines and reviews commonly applied neighborhood structures available in the literature of HFS problems. Moreover, this section describes several rules to select neighborhood operators and elaborates upon the feasibility of neighborhoods.

Besides that, Section 3.7 provides methods to deal with sequencing relations between the operations of a job. This section also describes techniques to extend these sequencing constraints to model, e.g., transportation times between operations, sequence-dependent changeovers between two consecutive jobs on the same machine, and release times for operations. Moreover, this section reviews several techniques to model sequencing constraints between different jobs. There are appears to be a gap in the literature regarding sequencing constraints between jobs in scheduling problems, which Afzalirad & Rezaeian (2017) also experience. Nevertheless, Section 3.7 outlines three methods to deal with feasibility regarding sequencing constraints, each with its benefits and drawbacks.

Finally, Section 3.8 reviews techniques to model limited additional resources (e.g., IBCs and operators). To the best of our knowledge, the work of Costa et al. (2020) and Tao et al. (2020) are the only studies that consider additional resources in an HFS environment. They propose a discrete backtracking search algorithm to solve this problem. In essence, this backtracking algorithm postpones jobs until the required resources are available. Nevertheless, there are some studies in other scheduling domains that consider additional resources. In other scheduling domains, most studies focus on a single resource type, thus, versions with multiple resource types are potential research areas (Costa, Fernandez-Viagas, & Framiñan, 2020). Moreover, there are almost no studies where the same additional resources can be shared over multiple different stages (e.g., IBC-filling, mixing, and packaging) simultaneously, as most studies only focus on shared resources within one stage (Edis, Oguz, & Ozkarahan, 2013).

4. Model alternatives

The main goal of this chapter is to answer the third research question:

"Which alternative models are suitable to solve the scheduling problem of Euroma?"

To achieve this, this chapter describes several modeling alternatives that we consider to solve our scheduling problem (see Section 2.6). We base these modeling alternatives on the findings of the literature review in Chapter 3 and our insights.

This chapter is organized as follows. Section 4.1 describes the assumptions and simplifications that we make. Section 4.2 describes the decisions of the problem that the model needs to consider. Section 4.3 describes the decoding algorithm that calculates the start- and finish times of operations in a schedule. Section 4.4 describes a corrective backtracking algorithm that ensures a maximum number of machine cleanings at the same time. Section 4.5 elaborates upon alternative objectives for our problem. Section 4.6 provides two alternative construction heuristics for our problem that we extend to ensure feasibility regarding the claim constraints. Subsequently, Section 4.7 elaborates upon two alternative neighborhood structures, their size, and connectedness. Moreover, this section describes two alternative improvement heuristics for our problem. Finally, Section 4.8 provides a summary of this chapter.

4.1 Model assumptions & simplifications

To simplify the modeling of the problem, we assume the following:

- All input data is deterministic, e.g., processing times, transportation times, and cleaning times.
- Jobs are always ready for production (e.g., all ingredients are available).
- Operators are always ready to assist.
- Machines do not have breakdowns.
- There is infinite storage capacity in the high-rise warehouse, which is located before the first stage and after the last stage.

4.2 Scheduling problem decisions

This section describes the decisions of the problem that the model needs to consider. Moreover, this section provides formal notations regarding the problem that the remaining sections in this chapter recall.

In essence, the scheduling model needs to make the following four decisions:

1. Allocate an eligible production route to every job;
2. Allocate an eligible machine to every operation of the allocated production route of a job;
3. Determine the sequence of the operations on the allocated machines;
4. Determine the start time of every operation.

Regarding the first decision, to be able to allocate an eligible production route to a job, we extend the general notation of the HFS problem as introduced in Section 3.7.1. The extension ensures that every job $j \in J$ needs processing according to any one of the eligible production routes in the set R_j, where the r^{th} route is denoted by $route_{j,r}$.

Regarding the second decision, every $route_{j,r}$ of job j consists of a sequence of consecutive operations $O_{j,r} = \{1,2,..,n_{j,r}\}$, where the i^{th} operation of job j on $route_{j,r}$ is denoted by $o_{j,r,i}$. Operation $o_{j,r,i}$ needs processing on any one of the eligible machines of the subset $M_{j,r,i} \subseteq M$.

Furthermore, recall from Section 2.2.2 that machines need maintenance occasionally. Often, the start of these jobs is somewhat flexible since Euroma has an in-house maintenance team. To be able to schedule maintenance, we let every maintenance job $w \in W$ have one $route_{w,1}$ and one operation $o_{w,1,1}$ with one eligible machine $k = M_{w,1,1}$. This resembles that maintenance job w is required on machine k. This way, it is possible to schedule maintenance jobs and regular jobs simultaneously; $J^{all} = J \cup W$.

Regarding the sequence of the allocated operations on a machine, the sequence must satisfy the claim constraints, as described in Section 2.2.2. Every job j contains a set of claims; $Claims = \{Halal, Kosher, \dots , Vegan\}$, where $c_{j,l} \in \{non-suitable, suitable, certified\}\ \forall l \in Claims$. It is not allowed to sequence an operation $o_{j,r,i}$ of which claim l is $c_{j,l} = certified$ within two positions after an operation $o_{j',r',i'}$ of which claim l is $c_{j',l} = non-suitable$ on the same machine $k \in M_{j,r,i} \cap M_{j',r',i'}$. Moreover, since Euroma is producing continuously, the new schedule should comply with the previous schedule. Thus, regarding the claim constraints, the first two operations in the sequence of machine k should comply with the last two operations of the previous sequence of machine k.

Regarding the fourth decision, we develop a decoding algorithm that determines the start time of every operation on the allocated machines. Section 4.3 describes the decoding algorithm.

4.3 Solution decoding algorithm

Once the sequences of operations on the machines are known and feasible regarding the claim constraints, it is possible to calculate the start- and finish times of every operation $o_{j,r,i}$. The start- and finish times of an operation $o_{j,r,i}$ are denoted by $s_{j,r,i}$ and $f_{j,r,i}$, respectively. This section describes the solution decoding algorithm that determines the start time of every operation by taking into account several constraints.

At first, recall that the operations of job j on route r have a predetermined sequence $O_{j,r}$. The operations $O_{j,r}$ should not overlap, i.e., $o_{j,r,i}$ cannot start earlier than $o_{j,r,i-1}$ is finished. Also, recall that a pair of consecutive operations $(o_{j,r,i-1}; o_{j,r,i})$ of a job j incurs a transportation time when transporting between two machines. Let $s^{job}_{j,r,i}$ be the minimum start time of the operation $o_{j,r,i}$ that is constrained by its operation sequence $O_{j,r}$.

Moreover, machines can only produce one operation at a time and there might be a sequence-dependent cleaning time between a pair of consecutive operations on the same machine. Therefore, denote $s^{mac}_{j,r,i}$ as the starting time of an operation constrained by its machine sequence. Note that every machine sequence should include the last operation of the previous schedule to avoid overlapping between the previous- and the new schedule. Besides, let $s^{rel}_{j,r,i}$ denote the minimum start time of the operation $o_{j,r,i}$ constrained by its release time. Also, recall that a machine cannot produce during one of its production-stops, as described in Section 2.2.2. Therefore, operations that are allocated to this machine cannot start or finish during one of the production-stops. To achieve this, we first calculate the minimum start time of operation $o_{j,r,i}$ as follows: $s_{j,r,i} \geq max\{s^{job}_{j,r,i}; s^{mac}_{j,r,i}; s^{rel}_{j,r,i}\}$.

Calculating the start times of the operations requires the allocation of operations to machines and the sequence of operations on the machines. The former is often represented by a machine-allocation-vector $\overrightarrow{V^m}$ and the latter is provided by a sequence-vector $\overrightarrow{V^s}$ (Li, et al., 2018). The sequence-vector consists of $\sum_{j \in J} n_j$ items, where n_j is the number of operations of job j, and each item stores one job number $j \in J$. Each job j appears n_j times in $\overrightarrow{V^s}$, where the first appearance of j resembles $o_{j,1}$, the second appearance of j resembles $o_{j,2}$, and so on. The machine- allocation-vector $\overrightarrow{V^m}$ has the same length as $\overrightarrow{V^s}$. In $\overrightarrow{V^m}$, each item stores one eligible machine number $k \in M_{j,i}$ of the operation $o_{j,i}$ that corresponds with the same item number in $\overrightarrow{V^s}$ (Li, et al., 2018).

For an example to calculate the start- and finish times of the operations, consider the operations in Table 13 that each need processing on one eligible machine $k \in M_{j,i}$ with a processing time $p_{j,i,k}$. Note that this instance considers no routes, cleaning-, transportation-, and release times.

Figure 25 provides a configuration of $\overrightarrow{V^s}$ and $\overrightarrow{V^m}$ for the instance in Table 13. We refer to this solution representation as encoded. To decode the solution, and thus, calculate the start- and finish times of the operations, follow the sequences in $\overrightarrow{V^s}$ and $\overrightarrow{V^m}$.

Table 13 | Processing times of the operations on eligible machines

$p_{j,i,k}$	k	1	2	3
$o_{j,i}$	$o_{1,1}$	1	1	2
	$o_{1,2}$		2	2
	$o_{2,1}$			2
	$o_{2,2}$	1	3	
	$o_{2,3}$		2	2
	$o_{3,1}$		1	3
	$o_{3,2}$	2	1	2

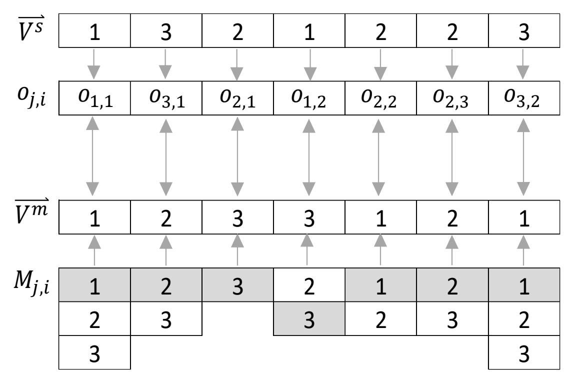


Figure 25 | Encoded solution representation

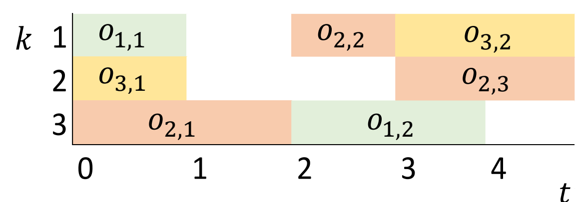


Figure 26 | Decoded solution representation

By following the sequence in $\overrightarrow{V^s}$ and $\overrightarrow{V^m}$, observe that machine 1 produces $o_{1,1}$ at $s_{1,1} = 0$, machine 2 produces $o_{3,1}$ at $s_{3,1} = 0$, and machine 3 produces $o_{2,1}$ at $s_{2,1} = 0$. At this point, every machine has one allocated operation. Continuing the sequence, machine 3 produces $o_{1,2}$ at $s_{1,2} = 2$, as $s_{1,2} \geq max\{s_{1,2}^{job} = 1; s_{1,2}^{mac} = 2; s_{1,2}^{rel} = 0\}$. Next, machine 1 produces $o_{2,2}$ at $s_{2,2} = 2$, since $s_{2,2} \geq max\{s_{2,2}^{job} = 2; s_{2,2}^{mac} = 1; s_{2,2}^{rel} = 0\}$. This process continues until all the start- and finish times of the operations are calculated. Figure 26 illustrates a Gantt chart of the decoded solution.

Recall that preemption of operations on machines is not allowed. Consequently, the finish time $o_{j,i}$ on machine k can be calculated by $f_{j,i} = s_{j,i} + p_{j,i,k}$. After calculating $s_{j,i}$ and $f_{j,i}$ of $o_{j,i}$, it is possible to verify whether $o_{j,i}$ starts or finishes during a production-stop. In such a case, we postpone the start time such that $o_{j,i}$ is not scheduled during a production-stop.

Note from Figure 26 that $o_{1,1}$ can postpone $s_{1,1}$ from 0 to 1 without delaying other operations. Similarly, $o_{3,1}$ can postpone $s_{3,1}$ from 0 to 2. This way, the flowtime of these jobs decreases, which enhances a stable usage of resources and less work-in-progress (Yenisey & Yagmahan, 2014). Therefore, we postpone the start time of an operation when it does not delay the finish time of any other operation.

4.4 Corrective backtracking algorithm

This section describes a corrective backtracking algorithm that ensures a maximum number of machine cleanings at the same time due to limited operator availability. To model this restriction, we use the idea of Afzalirad and Rezaeian (2017) to allow infeasibility and use a corrective backtracking algorithm to attain feasible solutions. We choose not to accept infeasible solutions by using a penalty to worsen the objective once a solution violates the constraint, as this might result in an infeasible final solution.

In essence, while computing the start times of the operations by iterating through the vectors $\overrightarrow{V^s}$ and $\overrightarrow{V^m}$, we list the start- and finish times for every cleaning in a list and sort this list in ascending order. This list refers to the value +1 for every start time, and -1 for every finish time. By iterating through this list, we enumerate the number of machine cleanings at every time a cleaning starts and finishes. Denote max_t^{clean} as the maximum number of allowed machine cleanings at time $t \in T$ due to limited operator availability. Once the number of machine cleanings at time $t \in T$ exceeds max_t^{clean}, the cleaning that was last added to the list gets postponed to the first feasible time such that max_t^{clean} is not exceeded.

Figure 27 illustrates a schedule where every pair of consecutive operations on the same machine requires a cleaning time of 3. Moreover, $max_t^{clean} = 1$ for $t \leq 10$, and $max_t^{clean} = 2$ otherwise. Note that a black bar indicates a cleaning and a colored bar indicates an operation $o_{j,r,i}$.

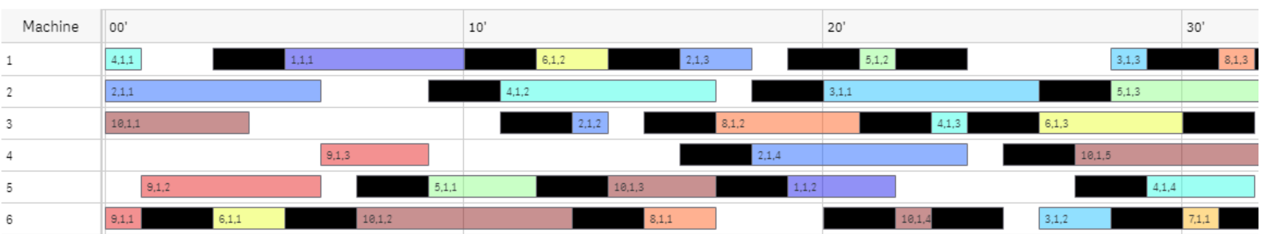


Figure 27 | Illustration of a schedule with a maximum number of machine cleanings allowed

Besides that, recall from Section 2.3.4 that the number of IBCs needed (i.e., in production and cleaning) cannot be more than the number of available IBCs, which is denoted by max^{IBC}. A corrective algorithm that modifies the start- and finish times of operations as we use to satisfy the maximum number of cleanings is insufficient to guarantee feasibility for the IBC-capacity constraint, as some sequences of $\overrightarrow{V^s}$ exceed max^{IBC} regardless of the start- and finish time of the operations. Hence, an algorithm is needed that can change $\overrightarrow{V^s}$ and determine the start- and finish times of the operations while still satisfying all remaining constraints. Such an algorithm is very computationally expensive. Therefore, we allow infeasibility and add a significant penalty ($IBC_{penalty}$) to worsen the objective once max^{IBC} is violated. To achieve this, let $usage_t^{IBC}$ be the number of IBCs in use at $t \in T$. Appendix 5 describes how we calculate the $usage_t^{IBC}$ at any time $t \in T$ based on the schedule. Finally, calculate the $IBC_{penalty}$ as follows:

$$IBC_{penalty} = \sum_{t \in T} Max(0, usage_t^{IBC} - max^{IBC}).$$

Note that when the $usage_t^{IBC}$ increases at $t \in T$, the $IBC_{penalty}$ also increases. Moreover, the $IBC_{penalty}$ increases when max^{IBC} is violated for a longer time. Therefore, this objective increases the $IBC_{penalty}$ depending on the degree of violation such that it is more likely to guide the search into the feasible region.

4.5 Objective function

This section elaborates upon suitable alternative objectives for our problem. These objectives are based on the knowledge obtained from the literature review and our insights. Moreover, as our problem has multiple objectives, this section describes which method we consider to deal with multiple objectives.

Euroma currently minimizes the total machine cleaning time CT_{tot}, since cleaning requires operators that cannot be assigned elsewhere when they are cleaning. Therefore, we include the CT_{tot} in the objective function. To achieve this, let $(o_{j',r',i'}; o_{j,r,i}) \in \Pi_k$ be a set of consecutive operation pairs in the sequence of machine k. The cleaning time between a pair of consecutive operations is denoted by $clean_{(j',r',i'),(j,r,i),(k)}$, where $k \in M_{j',r',i'} \cap M_{j,r,i}$ and $o_{j',r',i'} \neq o_{j,r,i}$. Note that cleaning can be required between the last operation of the previous sequence of machine k and the first operation of the new sequence of machine k. Therefore, $(o_{j',r',i'}; o_{j,r,i}) \in \Pi_k$ should also include the consecutive operation pairs, where $o_{j',r',i'}$ is the last operation of the previous sequence of machine k.

Additionally, according to the literature review in Section 3.4, there are three other suitable objectives for our problem. At first, minimizing the makespan C_{max} increases machine utilization and throughput (Yenisey & Yagmahan, 2014), which is in line with the main goal of Euroma. Second, minimizing the total tardiness T_{tot} provides a customer-centric approach (Tahmasbi & Moghaddam, 2011), which corresponds with the current objective of Euroma. In essence, T_{tot} calculates the total time that jobs are past their due-date d_j. Third, minimizing the total flowtime FT_{tot} enhances a stable usage of resources and less work-in-progress (Yenisey & Yagmahan, 2014). When the work-in-progress is low, mixtures stay less long in the IBCs, which favors the quality of the product. Moreover, minimizing the FT_{tot} might enhance a stable flow of IBCs through the process, which might help to satisfy the IBC-capacity constraint. Therefore, we also include the C_{max}, T_{tot}, and FT_{tot} in the objective function.

Recall that $s_{j,r,1}$ and $f_{j,r,n_{j,r}}$ are, respectively, the start time of the first operation and the finish time of the last operation of job j on $route_{j,r}$. Note that $s_{j,r,i} = 0$ and $f_{j,r,i} = 0$ when job j is not allocated to $route_{j,r}$. The formulations of the five aforementioned objectives are:

Maximum makespan: $C_{max} = max_{j \in J^{all}, r \in R_j} \left\{ f_{j,r,n_{j,r}} \right\};$

Total tardiness: $T_{tot} = \sum_{j \in J^{all}, r \in R_j} Max\left(0, f_{j,r,n_{j,r}} - d_j\right);$

Total flowtime: $FT_{tot} = \sum_{j \in J^{all}, r \in R_j} (f_{j,r,n_{j,r}} - s_{j,r,1});$

Total cleaning time: $CT_{tot} = \sum_{k \in M, (o_{j',r',i'}; o_{j,r,i}) \in \Pi_k} (cleaning_{(j',r',i'),(j,r,i),(k)});$

IBC violation penalty: $IBC_{penalty} = \sum_{t \in T} Max(0, usage_t^{IBC} - max^{IBC}).$

According to the literature review in Section 3.4, there are several multi-objective functions f. Minimizing the weighted sum of objectives (f_w) and the Pareto-optimal approach (f_p) are very common for multi objective scheduling problems according to the literature review of Yenisey and Yagmahan (2014, p. 132). We favor f_w over f_p since the latter requires selecting a good solution from a set of Pareto-optimal solutions every time a new schedule is required, as this might be necessary when the planners are not available. Moreover, f_p requires multiple solutions, which can be computationally expensive. Therefore, we select the objective function $Min\, f_w(Z_1, Z_2, \dots, Z_k)$, which minimizes the sum of the k weighted objectives. However, the drawback of this method is that it is difficult to determine the weights.

4.6 Construction heuristics

This section describes two alternative construction heuristics for our problem. At first, Section 4.6.1 describes a construction heuristic that randomly generates an initial solution for the purpose to benchmark the performance of the improvement heuristics. Second, Section 4.6.2 elaborates upon the NEH construction heuristic that Nawaz et al. (1983) propose and that Ruiz et al. (2008) extend for the HFS problem. We extend both construction heuristics such that they generate feasible solutions regarding the sequencing constraints (e.g., the claim constraints as described in Section 2.2.2). Only the limited IBC-capacity might not be satisfied, as exceeding the IBC-capacity is penalized in the objective function.

4.6.1 Random construction heuristic

The random construction heuristic (RCH) randomly generates initial solutions for the purpose to benchmark the improvement ability of the improvement heuristics. In essence, the RCH randomly selects a job that has at least one operation that still needs to be scheduled. For this job, the heuristic randomly selects an eligible production route. After that, the RCH selects the next operation of this route and randomly selects an eligible machine and a random position in the sequencing vector $\overrightarrow{V^s}$. The main challenge with generating an initial solution is to satisfy the claim constraints. The RHC verifies whether the claim constraints are satisfied. In the case that the claim constraints are not satisfied, the heuristic randomly selects another job, route, machine, and position. The heuristic may not find a feasible solution regarding the claim constraints after a certain number of iterations. In that case, the heuristic schedules two dummy operations at the end of the sequencing vector $\overrightarrow{V^s}$ and then it schedules the main operation after the dummy operations. The dummy operations resemble simple raw materials such as salt that can clean the pipes of the machines. Therefore, the heuristic always finds a feasible solution regarding the claim constraints. Appendix 6 provides a pseudo-code of the RCH.

4.6.2 Extended NEH construction heuristic

Ruiz et al. (2008) extend the NEH heuristic of Nawaz et al. (1983) for the HFS problem. Ruiz et al. (2008) show that the NEH heuristic is vastly superior to the other available construction heuristics. Liu, Yan, and Price (2017) extend the NEH heuristic by adding a tie-breaking rule. They show that their extension can find slightly better solutions for some problem instances.

We extend the work of Ruiz et al. (2008) such that the NEH heuristic always finds feasible solutions regarding the claim constraints. We choose not to focus on tie-breaking rules, as this might only result in a minor performance increase as reported in the literature (Liu, Jin, & Price, 2017). Instead, we choose to focus on obtaining a good improvement heuristic. In essence, the NEH heuristic consists of the following steps. First, list all jobs in a descending order based on their total average processing time ($TAPT_j$). However, as jobs may have different production routes, it is not possible to calculate the $TAPT_j$ by using the formula of Ruiz et al. (2008), as this formula ignores the production routes. Therefore, we extend the $TAPT_j$ formulation such that it takes into account the production routes. In essence, this formula calculates the average processing time to produce job j as follows:

$$TAPT_j = \frac{\sum_{r \in R_j, i \in O_{j,r}} \frac{\sum_{k \in M_{j,r,i}} p_{j,r,i,k}}{|M_{j,r,i}|}}{|R_j|}.$$

Second, take the first job of the list with the highest $TAPT_j$. Select a random route for this job and select the first operation that still needs scheduling. Place this operation into every feasible position in the sequencing vector $\overrightarrow{V^s}$ that does not violate the claim constraints and calculate the corresponding objective values. Select the feasible schedule with the best objective value. Subsequently, select the next operation of this jobs' route. Place this operation into every feasible position in the sequencing vector $\overrightarrow{V^s}$, without changing the relative positions of the already scheduled operations. Select the $\overrightarrow{V^s}$ with the best objective value. In the case that the claim constraints cannot be satisfied for a certain operation, the heuristic selects the next job on the list that still needs scheduling. When a feasible position is found, the search starts again with the first job on the list. Once all operations of a job are in the schedule, this job gets removed from the list. This is an iterative process that stops when there are no more jobs to schedule in the list.

In the case that there is no feasible position for the last job on the list, the heuristic schedules two dummy operations that are "suitable" for every claim at the end of the sequencing vector $\overrightarrow{V^s}$ and then it schedules the main operation after the dummy operations. The dummy operations resemble simple raw materials such as salt that can clean the pipes of the machines. Therefore, the heuristic always finds a feasible solution regarding the claim constraints. We do not consider scheduling dummy operations for operations other than the last job on the list that contains the jobs that still need to be scheduled, as there might be a feasible position for an operation of a job on the list that was not yet evaluated. Appendix 7 provides a pseudo-code of the extended NEH heuristic.

4.7 Improvement heuristics

Section 4.7.1 elaborates upon two alternative neighborhood structures that we consider as part of the improvement heuristics. Besides that, we consider three alternative improvement heuristics for our problem. Section 4.7.2 describes a simple improvement heuristic that does not accept worse solutions for the purpose to benchmark the performance of other more advanced improvement heuristics. Furthermore, following from the literature review in Section 3.5.2, we choose to select simulated annealing as a solution alternative as this heuristic shows promising results for HFS problems with sequence-dependent changeovers and transportation times between stages (Naderi, Zandieh, Khaleghei, & Roshanaei, 2009). Section 4.7.3 elaborates upon the simulated annealing heuristic.

Moreover, Section 4.7.4 describes an extension of the simulated annealing heuristic with tabu lists. We consider tabu lists as these can avoid going back and forth between neighbors. This characteristic might be beneficial for our problem since the claim constraints can limit the number of possible neighbor solutions, which increases the chance of evaluating a neighbor twice.

4.7.1 Neighborhood structures

This section elaborates upon the neighborhood structures that we consider for our problem. We select the following three neighborhood operators from the literature review in Section 3.6:

N1. Swap the operations $o_{j,r,i}$ and $o_{j',r',i'}$ on the same machine k;
N2. Move operation $o_{j,r,i}$ from position u to v on the same machine k;
N3. Move operation $o_{j,r,i}$ from position u to v from machine k to k', s.t. $k' \in M_{j,r,i}, k' \neq k$.

The operators (N1–N3) are commonly used for HFS problems and seem promising. Regarding their connectedness, N1 and N2 allow changing an initial sequence of a machine into any other sequence with the same operations in a finite number of iterations. Besides that, N3 allows moving operations between eligible machines.

Nevertheless, these neighborhood operators (N1–N3) cannot change the production route of a job. Therefore, we introduce the following neighborhood operator:

N4. Change the production route of job j from r to r', s.t. $r' \in R_j, r' \neq r$.

The operators (N1–N4) require selecting jobs, routes, operations, machines, positions in a sequence, or a combination of these. Most papers in the HFS field make these selections randomly and show promising results, e.g., the work of Naderi et al. (2009), Karimi et al. (2017), or Al-harkan and Qamhan (2019). We also choose to make all selections randomly.

Besides that, we consider two strategies to select neighborhood operators. The first strategy randomly selects any of the operators with equal probability. This strategy is commonly applied in the literature, e.g., the work of Karimi et al. (2017) or Zhang et al. (2019). The second strategy is inspired by the work of Naderi, Zandieh, and Roshanaei (2009) and makes a tradeoff between small operators (N1–N3) to intensify the search space, and conversely, a large operator (N4) to diversify the search space. In essence, we first intensively search the current search space by using the small neighborhood operators. After having searched the current space and the limit on the number of attempts for finding a better solution ($count_{stop}$) is reached, we use the operator N4 to identify a new search space. Appendix 8 describes the tuning process of the $count_{stop}$ parameter. Based on the results of this tuning process, we choose to set $count_{stop} = 5$.

All in all, the neighborhood operators (N1–N4) can (i) change for any job its production route, (ii) move any operation to any other eligible machine, and (iii) change any initial sequence of operations of a machine into any other sequence of operations. Moreover, the strategies for selecting a neighborhood operator may only exclude operators for a finite set of iterations. Thus, the neighborhood operator selection strategies can select every operator. Therefore, the neighborhood structure is connected as it can transform any initial solution into any other solution in a finite set of iterations.

4.7.2 Simple improvement heuristic

The simple improvement heuristic (SIH) is a basic solution alternative that only accepts better solutions than the current solution. Therefore, SIH is not able to escape from local optima. The performance of this heuristic allows benchmarking the performance of more advanced improvement heuristics that can escape from local optima.

The SIH heuristic starts with an initial solution S and sets this solution as the best solution S^*. SIH generates a neighbor solution S' from S^* and accepts S' if $F(S') < F(S^*)$. This process continues until either the time limit $time_{stop}$ is reached or the limit on the number of attempts for finding a better solution $count_{stop}$ is reached. Figure 28 provides a pseudo-code of the SIF heuristic. We set the following values for the SIH parameters: $time_{stop} = 150$ seconds and $count_{stop} = 10^5$.

Simple improvement heuristic (SIH)

Initialize $S^* \leftarrow S, count \leftarrow 0$
While Not $time_{stop}$ **and** $count < count_{stop}$ **do**
  $S' \leftarrow$ GenerateNeighbor(S^*)
  If $F(S') < F(S^*)$ **then**
    $S^* \leftarrow S'$
    $count \leftarrow 0$
  Else
    Increment($count$)
  EndIf
End
Return S^*

Figure 28 | Pseudo code simple improvement heuristic

4.7.3 Simulated annealing

Following from the literature review in Section 3.5.2, simulated annealing (SA) always accepts the neighbor solution S' when it is better than the current solution S. When the neighbor solution S' is worse than the current solution S, SA includes randomness in the acceptance criterion to be able to escape local optima (Ruiz & Vázquez-Rodríguez, 2010). In this case, the neighbor solution S' is accepted with a probability that depends on the difference between the objective values $F(S)$ and $F(S')$, and the progression of the heuristic, which is often denoted by the temperature T. The latter decreases over time with a factor α, which is referred to as the cooling factor. Therefore, the acceptance probability decreases over time such that it is less likely that SA accepts worse solutions over time. The heuristic stops when $T \leq T_{stop}$. Finally, SA returns the best solution found S^*. Figure 29 provides pseudo code of SA.

Simulate annealing

Initialize: $S^* \leftarrow S, T \leftarrow T_{start}, T_{stop}, MarkovLen, \alpha$
While $T > T_{stop}$ **do**
  For $k = 1$ **to** $MarkovLen$ **do**
    $S' \leftarrow$ GenerateNeighbor(S)
    If $F(S') < F(S^*)$ **then**
      $S^* \leftarrow S'$
    EndIf
    If $F(S') \leq F(S)$ **or** $\exp\left(-\frac{F(S)-F(S')}{T}\right) < \text{random}(0,1)$ **then**
      $S \leftarrow S'$
    EndIf
  End
  $T \leftarrow \alpha T$
End
Return S^*

Figure 29 | Simulated annealing pseudo code

Appendix 9 describes the tuning process of the SA parameters. Based on the results of this tuning process, we choose to set the following SA parameter values: $T_{start} = 100$, $T_{stop} = 0.1$, $MarkovLen = 750$, and $\alpha = 0.99$.

4.7.4 Simulated annealing with a tabu list

Following from the literature review in Section 3.5.2, tabu lists have the advantage to avoid going back and forth between neighbors. This characteristic might be beneficial for our problem since the claim constraints can limit the number of possible neighbor solutions, which increases the chance of evaluating a neighbor twice. Moreover, decoding a neighbor solution to obtain the objective value is computationally expensive for our problem, so evaluating a neighbor twice can be considered computational waste. Therefore, we extend the simulated annealing heuristic as described in Section 4.7.3 with a tabu list.

There are several possible tabu lists, e.g., storing the complete neighbor solutions visited, the objective values of the visited solutions, or the neighborhood operations performed. Dauzère-Pérès et al. (2018) show the efficiency of a tabu list that stores the recent neighborhood operations performed. Suggesting a neighborhood operation occurs before decoding a solution and calculating the objective value, which might enhance the computational performance. Moreover, verifying whether a neighborhood operation is in the tabu list is less computationally expensive than verifying whether a complete solution is in the tabu list. Furthermore, storing the recent neighborhood operations performed can be done efficiently per neighborhood operator. Therefore, we opt for a tabu list that stores the recent neighborhood operations performed. However, a possible drawback of this might be that a neighbor solution that has never been visited gets rejected by the tabu list since the neighborhood operator is in the tabu list. We choose not to use aspiration criteria to avoid this issue, since such aspiration criteria require storing the complete neighbor solution, which increases the computational time.

We choose to have a tabu list for each neighborhood operator since this allows to set the length of the tabu list per operator and it is computationally more efficient since it is possible to immediately select the list of tabu neighborhood operators that are applicable. The tabu lists that we consider are as follows:

N1. Swap $o_{j,r,i}$ with $o_{j',r',i'}$ on the same machine: Swap.Add($o_{j',r',i'}; o_{j,r,i}$);
N2. Move $o_{j,r,i}$ on the same machine k from position u to v: Move.Add($o_{j,r,i};\ k;\ u$);
N3. Move $o_{j,r,i}$ from machine k to k' from position u to v, s.t. $k' \in M_{j,r,i}, k' \neq k$: Move.Add($o_{j,r,i};\ k;\ u$);
N4. Change the production route of job j from r to r', s.t. $r' \in R_j, r' \neq r$: Change.Add($j; r$).

Note that three tabu lists are sufficient, as the move operations can be stored in one tabu list. Besides that, note that the stored operations are the reversal of the executed operations.

Appendix 10 describes the tuning process of the length for each tabu list. Based on the results of this tuning process, we choose to set the length of the swap and move tabu lists at 75 and the length of the change production route tabu list at 50. Note that the length of the change production route tabu list is lower than the length of the swap and move tabu lists. This might be due to the limited number of neighbors for the change route operator compared to the other operators.

4.8 Summary of the model alternatives

The main goal of this chapter is to answer the third research question: Which alternative models are suitable to solve the scheduling problem of Euroma? To answer this question, we describe the assumptions in Section 4.1. Section 4.2 first outlines the decisions of the problem that the model needs to consider. The main difference between our problem and HFS problems in the literature is that our problem considers multiple eligible production routes per job, whereas problems in the literature generally consider a single production route per job. Besides that, this section describes how we combine the maintenance jobs with the regular production jobs during scheduling. Moreover, this section explains the claim constraints.

Section 4.3 explains a decoding algorithm that determines the start- and finish time of every operation. This decoding algorithm takes into account the following characteristics and constraints:

- Operations of a job cannot overlap and need to produce in a predetermined sequence;
- A pair of consecutive operations of a job incurs a transportation time between two machines;
- Jobs should not start earlier than their release time;
- Machines can only produce one operation at a time and there might be a sequence-dependent cleaning time between a pair of consecutive operations on the same machine;
- Preemption of operations on machines is not allowed;
- A machine cannot produce during one of its production-stops;
- The new schedule should comply with the previous schedule.

Section 4.4 describes a corrective backtracking algorithm that ensures a maximum number of machine cleanings at the same time due to limited operator availability. In essence, this algorithm computes the number of machine cleanings at any time a cleaning starts and finishes. In the case that the number of allowed machine cleanings is exceeded, the algorithm postpones the machine cleaning that was last added to the list such that this constraint is satisfied.

Section 4.5 explains why we select the objective function $Min\ f_w(C_{max}, CT_{tot}, T_{tot}, FT_{tot}, IBC_{penalty})$, which minimizes the sum of the five weighted objectives. The drawback of this method is that it is difficult to determine the weights.

Section 4.6 describes two alternative construction heuristics for our problem. At first, this section describes a construction heuristic that randomly generates an initial solution for the purpose to benchmark the performance of the improvement heuristics. Second, this section elaborates upon the NEH construction heuristic that Nawaz et al. (1983) propose and that Ruiz et al. (2008) extend for the HFS problem. We extend both construction heuristics such that they always generate feasible solutions regarding the sequencing constraints (e.g., the claim constraints). Only the limited IBC-capacity might not be satisfied, as exceeding the IBC-capacity is a soft constraint that is penalized in the objective function.

Section 4.7 first describes three common neighborhood operators for HFS problems. We introduce a new neighborhood operator that can change the production route of a job. Moreover, this section describes two alternative neighborhood structures. The first neighborhood structure randomly selects any of the operators with equal probability. The second neighborhood structure makes a tradeoff between small operators to intensify the search space, and conversely a large operator to diversify the search space.

Besides that, this section describes three alternative improvement heuristics for our problem. At first, a simple improvement heuristic that does not accept worse solutions for the purpose to benchmark the performance of other more advanced improvement heuristics. Furthermore, we choose to select simulated annealing as a solution alternative as this heuristic shows promising results for HFS problems with sequence-dependent changeovers and transportation times between stages (Naderi, Zandieh, Khaleghei, & Roshanaei, 2009). Furthermore, following from the literature review, tabu lists have the advantage to avoid going back and forth between neighbors. This characteristic might be beneficial for our problem since the claim constraints can limit the number of possible neighbor solutions, which increases the chance of evaluating a neighbor twice. Moreover, decoding a neighbor solution to obtain the objective value is very computationally expensive for our problem, so evaluating a neighbor twice can be considered computational waste. Therefore, we extend the simulated annealing heuristic with the tabu list property.

5. Experiments

The main goal of this chapter is to answer the third research question:

"Which alternative solution approach performs best compared to the current situation under different experimental settings?"

To achieve this, Section 5.1 describes the problem instances of the company data that we use for the experiments. At first, we aim to identify the most promising model configuration (i.e., the construction- and improvement heuristics including the corresponding parameter sets that result in the best solutions according to the objective function). Section 5.2 describes the experimental design to identify the most promising model configuration among the possible model configuration alternatives (see Chapter 4). Second, Section 5.2 describes the experimental design to compare the performance of the selected model with the simulated performance of the current situation.

Regarding the experimental results, Section 5.3 provides the results of the different model configurations and Section 5.4 provides the results of the comparison between the performance of the selected model with the simulated performance of the current situation. Finally, Section 5.4 evaluates the performance of a simple version of the model after implementation in practice.

5.1 Problem instances

This section provides a summary of the problem instances that are extracted from the company data in 2021 that we use for the experiments. At first, the data of Euroma currently lacks the eligible production routes per job. Therefore, Appendix 12 explains how we obtain the eligible production routes per job based on historical data. Appendix 13 describes how we obtain the processing times of jobs on machines. Furthermore, Euroma currently lacks a contamination matrix. Hence, Appendix 14 describes the configuration of the contamination matrix based on data and results from the laboratory.

As Euroma aims to grow its production demand, the model should be able to cope with different demand volumes. Therefore, we consider problem instances with low, normal, and high demand. Table 14 provides the instance-specific job information, of which the standard deviation is reported between brackets.

Table 14 | Instance-specific job information

Instance	Demand	Jobs	Routes per job	Operations per route	Machines per operation	Processing time	IBCs replenish	IBCs discharge	Release dates	Due dates
1	Low	207	3.4 (1.9)	2.5 (0.6)	2.2 (1.5)	90.0 (44.1)	2.2 (1.2)	1.7 (1.4)	6	10
2	Low	197	3.6 (2.0)	2.6 (0.6)	2.1 (1.5)	87.1 (46.2)	2.2 (1.3)	1.8 (1.5)	8	8
3	Normal	305	3.8 (2.0)	2.6 (0.5)	2.1 (1.5)	85.6 (43.6)	2.1 (1.2)	1.7 (1.3)	9	8
4	Normal	302	3.3 (2.0)	2.6 (0.6)	2.1 (1.5)	85.0 (47.3)	2.1 (1.2)	1.7 (1.4)	16	10
5	High	403	3.6 (2.0)	2.6 (0.6)	2.1 (1.5)	86.1 (46.4)	2.1 (1.2)	1.7 (1.4)	10	11
6	High	404	3.6 (2.0)	2.6 (0.5)	2.1 (1.5)	84.3 (45.5)	2.0 (1.2)	1.6 (1.3)	14	15

The low demand instances 1 and 2 are based on 70% of randomly selected production jobs of the company data from weeks 16 and 17, respectively. The normal demand instances 3 and 4 are extracted from the company data from weeks 18 and 19, respectively. The high demand instance 5 includes all production jobs of week 20 and 30% of randomly selected jobs of week 19. Instance 6 includes all production jobs of week 21 and 30% of randomly selected jobs of week 20.

Furthermore, Table 16 provides the cleaning time distribution per instance. Moreover, Table 15 provides the percentages of jobs that are certified, suitable, and non-suitable for a claim.

Table 16 | Instance-specific cleaning duration information

	Cleaning duration (minutes)					
Instance	0	10	30	45	75	150
1	32.7%	3.8%	45.9%	0.5%	11.6%	5.5%
2	38.4%	6.1%	39.7%	0.8%	11.4%	3.6%
3	38.2%	7.5%	37.0%	1.0%	13.2%	3.2%
4	35.5%	6.7%	42.1%	0.8%	12.1%	2.8%
5	34.9%	5.3%	41.9%	0.7%	14.5%	2.9%
6	37.1%	8.2%	38.1%	0.7%	13.3%	2.6%

Table 15 | Instance-specific claim information

	% Of jobs with claim values		
Instance	Certified	Suitable	Non-suitable
1	9.2%	71.0%	19.8%
2	8.4%	76.6%	15.0%
3	5.1%	73.3%	21.6%
4	8.6%	73.8%	17.5%
5	8.1%	69.6%	22.3%
6	6.2%	72.0%	21.8%

Recall that the new schedule must comply with the previous schedule as the factory is producing continuously (see Section 2.2.2). Also, recall that we denote Π_k^{Prev} as the previous sequence of jobs on machine k and f_k^{Prev} as the finish time of Π_k^{Prev}. The claim constraints require that the first two jobs in the sequence of machine k should comply with the last two jobs of the previous sequence Π_k^{Prev} of machine k. Therefore, we set the last two jobs of machine k of the previous week of the data instance as Π_k^{Prev}, and f_k^{Prev} as the finish time of the last job on machine k. Moreover, we set $IBC^{prev} = 10$, which is the average number of IBCs in cleaning at the time $max_{k \in M}\{f_k^{Prev}\}$, which is extracted from the ESA IBC-log over the period 05-01-2020 until 30-04-2021.

Recall that max_t^{clean} is the maximum allowed number of mixer cleanings at time $t \in T$. For every problem instance, we set max_t^{clean} at 4 for all $t \in T$. Moreover, we set max^{IBC} at 60, which is the maximum allowed number of IBCs needed (i.e., in production and cleaning) at any time.

As there was no maintenance on the machines in the weeks 16-21 of 2021, we do not add extra maintenance jobs. Moreover, as there currently is a production backlog, there were no production-stops on any of the machines in weeks 16-21. Therefore, we also do not consider production-stops.

We set $transp_{k,k'}$, which is the time to transport an IBC or a big bag between any two machines $k \in M_{j,r,i}$ and $k' \in M_{j,r,i+1}$ at 15 minutes. Furthermore, we set the time to fill a machine with one IBC at $filltime_k^{IBC} = 15$ minutes for all $k \in M$. Moreover, we set for each of the two IBC-cleaning stations a cleaning time of $cleaning^{IBC} = 16$ minutes per IBC. The data instances are validated by the process engineer, production manager, and the business analyst of Euroma. According to these experts, the data instances provide a good representation of reality.

5.2 Experimental design

This section describes the experimental design, experimental settings, and the KPIs to evaluate the performance of the models. At first, we aim to identify the most promising model configuration (i.e., the construction- and improvement heuristics including the corresponding parameter sets that result in the best solutions according to the objective function). Section 5.2.1 describes the experimental design to identify the most promising model configuration. Section 5.2.2 describes the experimental design to compare the performance of the selected model with the simulated performance of the current situation. This section also describes the experiment to evaluate a simple version of the model in practice.

We implement all experiments in Delphi 10.3 on a Windows machine equipped with an Intel Hexa-Core 4.1GHz processor and 16GB of RAM.

5.2.1 Experiments on the alternative model configurations

This section provides the experimental design to identify the most promising model configuration. Recall that Section 4.5 describes why we consider the five objectives C_{max}, T_{tot}, CT_{tot}, FT_{tot} and $IBC_{penalty}$. Also, recall that we select the objective function $Min\ f_w$, which minimizes the sum of the five weighted objectives. Experiment 1, which we refer to as Exp1, describes how we obtain the weight set of f_w. Exp2 evaluates the performance of the alternative model configurations.

Experiment 1 | Objective weight tuning

This experiment tunes the weights of the five objectives to identify which weight set is most suitable for Euroma. We use the Pareto-optimal method with an a posteriori approach (as described in Section 3.4) in which we iteratively tune the weight set with experts from Euroma.

In this experiment, we select the heuristics NEH and simulated annealing (SA) to solve problem instance 3. This instance has 305 jobs, which corresponds with the current demand of Euroma, as described in Section 1.2. The neighborhood structure of SA in these experiments consists of the four neighborhood operators (N1–N4) which are selected randomly, as described in Section 4.7.1. Appendix 9 describes the tuning process of the SA parameters. Based on the results of this tuning process, we choose to set the following SA parameter values: $T_{start} = 100$, $T_{stop} = 0.1$, $MarkovLen = 750$, and $\alpha = 0.99$.

To tune the weight sets, we start with a set that results in decent quality solutions. We use a dashboard (see Section 6.2) to evaluate the quality of the solutions by reviewing the KPIs C_{max}, T_{tot}, CT_{tot}, the average buffer time of a job (BT_{avg}), and the maximum number of IBCs needed (IBC_{max}). We also report the number of feasible solutions regarding the IBC-capacity constraint. Based on these results, we determine together with Euroma the next weight set to evaluate. For each set, we conduct 10 replications (i.e., solving every scenario 10 times with the same parameter settings to provide statistically significant results). We choose to perform 10 replications as this results in a total CPU time of 6 hours for this experiment, which fits in a regular working day. We perform these replications for two reasons (i) due to the randomness of the heuristics and (ii) to obtain the variability of the single objective values in the weighted objective function.

We select a suitable weight set and evaluate this set for all problem instances. Subsequently, we fix this set for the remaining experiments. Fixing the weight set is necessary to be able to construct good solutions without human interaction, since the planners are not available continuously to run and tune the model.

Experiment 2 | Evaluating the alternative model configurations

This experiment evaluates the performance of the alternative model configurations that each consist of (i) a construction heuristic, (ii) a neighborhood structure, and (iii) an improvement heuristic. Regarding the construction heuristics, we consider the random construction heuristic (RCH) for the purpose to benchmark the performance of the improvement heuristics, and the NEH construction heuristic (NEH).
We consider two neighborhood structures, as described in Section 4.7.1. One neighborhood structure selects the four neighborhood operators (N1 – N4) randomly, whereas the other neighborhood structure uses a strategy to select neighborhood operators. Section 4.7.1 describes the parameter values of the neighborhood structures. We refer to the random neighborhood structure as RN and to the neighborhood structure that uses a strategy as SN.

Regarding the improvement heuristics, we consider the simple improvement heuristic (SIH), simulated annealing (SA), and SA with tabu lists (SATL), as described in Section 4.7. Recall that the SIH heuristic only accepts better solutions than the current solution. Therefore, SIH is not able to escape from local optima. The performance of this heuristic allows benchmarking the performance of SA and SATL, which can escape from local optima. The parameter values of the improvement heuristics are described in Section 4.7.

For each model configuration that consist of (i) a construction heuristic, (ii) a neighborhood structure, and (iii) an improvement heuristic, this experiment conducts 10 replications for each problem instance and reports the KPIs. We also report the objective values, the number of feasible solutions regarding the IBC-capacity constraint, and the computational time in seconds (CPU). After evaluating the performance of the model configurations, we select the configuration that provides good quality solutions regarding the objective function in short computational times for the remaining experiments.

5.2.2 Experiments to evaluate the impact of the model

This section provides the experimental design of the experiments Exp3 – Exp6 to evaluate the performance of the model compared to the current situation.

Experiment 3 | Evaluate the effect of optimizing the schedules of the stages simultaneously

This experiment evaluates the effect of optimizing all stages simultaneously compared to the current situation of Euroma, which optimizes the schedules separately. Recall that Euroma currently first optimizes the mixing schedule to minimize both CT_{tot} and T_{tot}, as described in Section 1.3.1. Subsequently, ESA allocates the IBC-filling operations to the first IBC-filling station that becomes available, without changing the relative order of operations in the mixing schedule, as described in Section 2.3.2. In essence, ESA minimizes the flowtime FT_{tot}. After that, the discharging schedule is optimized by minimizing the CT_{tot}, T_{tot}, and FT_{tot}, without changing the mixing schedule. During this process, the production routes of the jobs remain unchanged.

We simulate the current scheduling process of Euroma by only considering the default production routes and by scheduling in the following order:

1. Optimize the mixing schedule by minimizing CT_{tot}, T_{tot};
2. Optimize the IBC-filling schedule by minimizing FT_{tot}, without changing the relative order of operations in the mixing schedule;
3. Optimize the discharging schedule by minimizing CT_{tot}, T_{tot}, and FT_{tot}, without changing the relative order of operations in the IBC-filling- and mixing schedules.

For each scheduling step, we use the selected model configuration to generate the schedule that corresponds to that step. We simulate the simultaneous optimization of all three stages by using the selected model configuration, without changing the default production routes. For each problem instance, this experiment conducts 25 replications and reports the KPIs for both the separate- and the simultaneous optimization of the stages. For the remaining experiments, we choose to perform 25 replications instead of the 10 replications as performed in Exp1 and Exp2, since we aim to obtain more statistically significant results to evaluate the impact of the model.

Experiment 4 | Evaluate the effect of changing the production routes
The model allows changing the production routes of jobs. This experiment evaluates the effect of the ability to change the production routes compared to the current situation of Euroma, which only considers the default production routes. We consider two scenarios that allow (i) only the default production route and (ii) changing the production route to any other eligible route. For both scenarios and each problem instance, this experiment conducts 25 replications and reports the KPIs.

Experiment 5 | Compare the performance of the current situation to the performance of the model
This experiment compares the performance of the current situation of Euroma to the performance of the selected model. To achieve this, we first simulate the current scheduling process of Euroma, as described in EXP3. We compare this performance to the performance of the model that can optimize all stages simultaneously and can change the production routes of the jobs. For both scenarios and each problem instance, this experiment conducts 25 replications and reports the KPIs.

Experiment 6 | Evaluate the performance of a simple version of the model in practice
To evaluate the performance of the model in practice, we implement a simple version of the model in practice to optimize the mixing schedule by minimizing CT_{tot} and T_{tot}. This model only considers the default production routes. The model uses the NEH and SA heuristics with only the move and swap operators (N1 and N2), which are selected randomly. The parameters of SA are described in Appendix 9. Essentially, this model replaces the need for the production manager to create the mixing schedules.

This experiment evaluates the performance in practice of the model that only optimizes the mixing schedule compared to the current situation. To achieve this, the model generates a mixing schedule for each week for 20 consecutive weeks. The mixing department produces according to these mixing schedules. Unfortunately, the CT_{tot} and T_{tot} are not registered in practice. Therefore, it is only possible to report the T_{tot} of the model solutions and the CT_{tot} based on the contamination matrix. We compare these results to the period before implementing the model.

5.3 Results of the alternative model configurations

This section provides the experimental results of the different model configurations, as described in Section 5.2.1. Regarding the experimental results, Section 5.3.1 provides the results of Exp1 on the objective weight tuning and Section 5.3.2 provides the results of Exp2 on the alternative model configurations. Regarding the results, the timestamps have the format "d:hh:mm:ss".

5.3.1 Results of the objective weight tuning

This section provides the results of the objective weight tuning, as described in Section 5.2.1. At first, Exp1 tunes the weight set of the objective function by using problem instance 3. Based on these results, we select a suitable weight set. Subsequently, we evaluate this weight set for all problem instances.

Experiment 1 | Objective weight tuning
This experiment tunes the weights of the objective function $Min\ f_w(C_{max}, T_{tot}, CT_{tot}, FT_{tot}, IBC_{penalty})$ to identify which weight set is most suitable for Euroma. We use the Pareto-optimal method with an a posteriori approach in which we iteratively tune the weight set with experts from Euroma, as described in Section 5.2.1. Table 17 provides the weight sets $\{w_{C_{max}}, w_{T_{tot}}, w_{CT_{tot}}, w_{FT_{tot}}, w_{IBC_{penalty}}\}$ per scenario including the resulting KPIs of instance 3, of which the standard deviation is reported between brackets.

Table 17 | Objective weight set tuning results

Scn	$w_{C_{max}}$	$w_{T_{tot}}$	$w_{CT_{tot}}$	$w_{FT_{tot}}$	w_{IBC}	C_{max}	T_{tot}	CT_{tot}	BT_{avg}	IBC_{max}	Feasible	CPU
1	13%	14%	13%	30%	30%	5:05:17:00 (2:44:27)	0	7:01:41:00 (2:42:08)	3:13:36 (18:05)	45.6 (2.9)	100%	146.9 (0.8)
2	12%	14%	14%	30%	30%	5:04:18:00 (4:33:16)	0	6:16:43:00 (3:57:47)	3:34:34 (22:18)	47.8 (4.0)	100%	152.7 (11.6)
3	11%	14%	15%	30%	30%	5:08:25:00 (3:14:07)	0	6:18:41:00 (2:59:04)	3:34:29 (8:20)	50.4 (5.5)	100%	142.6 (1.0)
4	14%	14%	12%	30%	30%	5:05:09:00 (2:12:09)	0	7:02:16:00 (6:46:09)	3:10:16 (26:29)	47.2 (4.3)	100%	146.6 (8.0)
5	15%	14%	11%	30%	30%	5:03:39:00 (2:59:54)	0	7:07:02:00 (5:58:28)	3:28:46 (33:16)	44.8 (2.3)	100%	143.1 (3.6)
6	16%	14%	10%	30%	30%	5:04:42:00 (2:36:28)	0	7:08:41:00 (4:54:04)	2:58:16 (19:06)	48.2 (2.8)	100%	158.9 (6.6)
7	12%	14%	12%	32%	30%	5:07:27:00 (3:23:40)	0	6:23:47:00 (6:05:47)	3:00:20 (27:43)	45.8 (5.6)	100%	147.1 (6.7)
8	14%	14%	14%	28%	30%	5:03:15:00 (2:28:27)	0	6:15:19:00 (4:46:55)	3:28:15 (17:53)	48.4 (2.2)	100%	154.1 (1.1)
9	15%	14%	15%	26%	30%	5:03:37:00 (2:07:33)	0	6:18:54:00 (4:41:06)	4:04:58 (30:15)	53.2 (3.6)	100%	155.1 (6.8)
10	12%	14%	12%	30%	32%	5:05:48:00 (2:37:24)	0	6:23:38:00 (3:04:17)	3:24:14 (14:15)	46.5 (2.1)	100%	145.5 (1.1)
11	14%	14%	14%	30%	28%	5:05:25:00 (2:43:55)	0	7:01:13:00 (2:56:48)	3:15:52 (15:35)	45.9 (1.6)	100%	148.2 (2.5)
12	15%	14%	15%	30%	26%	5:05:56:00 (2:14:12)	0	7:00:18:00 (2:45:51)	3:33:26 (19:13)	47.2 (3.1)	100%	147.4 (1.6)

Following from Table 17, the results of the weight set scenarios are as follows:

- Scenarios 1-3 shift the weight from $w_{C_{max}}$ to $w_{CT_{tot}}$, resulting in a higher C_{max} and a lower CT_{tot};
- Scenarios 4-6 shift the weight from $w_{CT_{tot}}$ to $w_{C_{max}}$, resulting in a higher CT_{tot} and a lower C_{max};
- Scenarios 7-9 shift the weight from $w_{FT_{tot}}$ to both $w_{C_{max}}$ and $w_{CT_{tot}}$, resulting in a higher BT_{avg} and a lower C_{max} and CT_{tot};
- Scenarios 10-12 shift the weight from w_{IBC} to both $w_{C_{max}}$ and $w_{CT_{tot}}$, resulting in similar results as scenario 1. This is logical since the $IBC_{penalty}$ only gets a value once the IBC-capacity is exceeded. For this instance, the IBC-capacity is not exceeded as all solutions are feasible. Therefore, scenarios 1 and 10-12 show similar results.

From the results in Table 17, we choose to select the weight set $\{14\%, 14\%, 14\%, 28\%, 30\%\}$ of scenario 8. This weight set results in a low C_{max} and CT_{tot}, while still having an IBC_{max} that does not get too close to the IBC-capacity. An IBC_{max} that is close to the IBC-capacity might result in infeasible solutions for other problem instances.

Next, we evaluate whether the selected weight set is appropriate for all other problem instances. Appendix 15 provides the results of 10 replications per problem instance. These results show that the selected weight set provides feasible solutions for all problem instances over all replications. Therefore, we select this weight set for the remaining experiments.

5.3.2 Evaluating the alternative model configurations

This section provides the experimental results of the alternative model configurations, as described in Section 5.2.1. Based on these results, we select the configuration that provides good quality solutions regarding the objective function in short computational times for the remaining experiments.

Experiment 2 | Results of the alternative model configurations

Recall that a model configuration consists of (i) a construction heuristic, (ii) a neighborhood structure, and (iii) an improvement heuristic. Table 18 provides the experimental results of the KPIs for each model configuration over all problem instances, sorted based on the average improvement heuristic objective value in ascending order. In the column that reports the objective values (Obj), the first value corresponds with the construction heuristic and the second value with the improvement heuristic. The standard deviations are reported between brackets. Note that the standard deviations are relatively high due to the variability in the demand of the problem instances. Appendix 16 contains the detailed results per instance.

Table 18 | Model configuration performance results

Configuration	Obj	C_{max}	T_{tot}	CT_{tot}	BT_{avg}	IBC_{max}	Feasible	CPU
NEH, SA, RN	21843	5:01:50:48	0	6:08:17:25	3:28:18	47.8	100%	135.5
	7064	(1:03:50:54)		(2:11:27:40)	(1:00:25)	(5.6)		(42.7)
RCH, SA, RN	69658	5:02:36:04	0	6:08:34:20	3:29:12	47.8	97%	131.2
	7076	(1:04:11:25)		(2:11:08:05)	(56:35)	(5.9)		(40.3)
NEH, SA, SN	21709	5:02:38:37	0	6:08:24:20	3:29:29	47.7	97%	145.0
	7076	(1:03:33:46)		(2:10:36:16)	(57:31)	(6.1)		(42.1)
RCH, SA, SN	69872	5:02:10:54	0	6:09:13:00	3:30:36	48.5	97%	146.4
	7079	(1:03:52:54)		(2:10:43:07)	(54:32)	(5.7)		(41.6)
RCH, SATL, RN	71334	5:03:36:23	0	6:12:36:50	3:43:43	49.0	95%	115.3
	7209	(1:03:46:33)		(2:12:17:49)	(1:03:19)	(7.3)		(35.9)
NEH, SATL, RN	22012	5:04:36:00	0	6:14:36:05	3:47:47	49.6	92%	120.6
	7284	(1:03:51:45)		(2:12:56:10)	(1:01:27)	(6.8)		(37.9)
RCH, SATL, SN	70845	5:01:48:20	0	6:17:03:55	5:04:38	57.5	68%	116.0
	7468	(1:02:56:57)		(2:12:32:23)	(1:11:07)	(7.0)		(44.4)
NEH, SATL, SN	21991	5:02:17:42	0	6:18:52:00	5:04:03	57.3	63%	131.4
	7505	(1:03:22:22)		(2:12:25:57)	(1:12:51)	(8.1)		(29.1)
NEH, SIH, RN	21388	7:07:22:42	21:09:51:27	9:09:30:55	1:02:10:23	175.5	0%	15.7
	21385	(1:18:58:20)	(8:17:14:05)	(2:06:18:36)	(6:20:18)	(47.8)		(6.6)
NEH, SIH, SN	21530	7:10:27:03	21:10:04:51	9:07:22:55	1:03:10:00	173.3	0%	17.3
	21528	(1:21:51:27)	(8:01:01:03)	(2:02:54:38)	(6:41:29)	(50.3)		(7.5)
RCH, SIH, SN	69669	16:17:34:47	69:16:33:10	17:22:09:10	6:09:06:19	508.3	0%	150.0
	65892	(5:06:52:10)	(43:07:52:55)	(4:21:08:18)	(1:22:25:29)	(134.6)		(0.0)
RCH, SIH, RN	70166	16:22:29:33	75:02:47:50	18:02:09:20	6:10:49:50	513.0	0%	146.9
	68069	(5:11:25:39)	(46:16:45:08)	(4:20:08:22)	(1:23:16:23)	(138.3)		(18.8)

Following from the results in Table 18, we observe the following:

- The NEH construction heuristic outperforms the RCH. Nevertheless, the initial solution seems to have no influence on the final objective value for the improvement heuristics SA and SATL. This indicates that SA and SATL both have a good improvement ability, regardless of the initial solution.
- Recall that both SA and SATL can escape from local optima and SIH cannot escape from local optima. The results show that both SA and SATL outperform SIH. This indicates that the improvement heuristic should be able to escape from local optima to obtain good solutions.

- The results show that SA obtains more feasible solutions than SATL regarding the IBC-capacity constraint. The SIH heuristic is not able to obtain feasible solutions.
- The results indicate that SA outperforms SATL regarding the objective. A possible explanation of this might be that SATL could reject neighbors that have never been visited since the neighborhood operator is in the tabu list. Therefore, SATL might not be able to escape its local optimum.
- The results indicate that selecting neighborhood operators randomly results in better solutions for the SATL heuristic than using a strategy to select neighborhood operators.

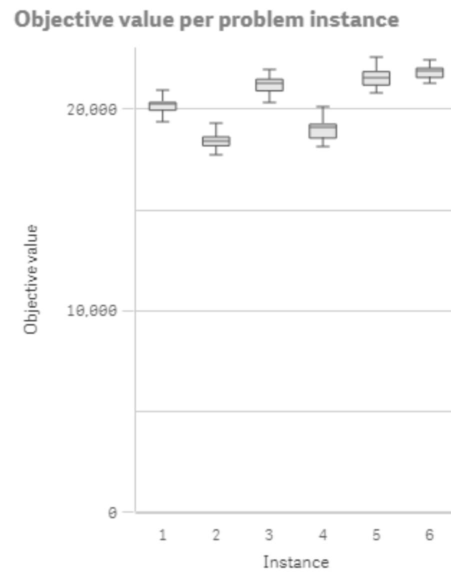


Figure 30 | Objective value per problem instance over 25 replications (NEH, SA, RN)

The results indicate that the configuration (NEH, SA, RN) results in the best average objective value while still obtaining 100% feasibility. Moreover, we observe from Figure 30 that there is relatively low variation in the objective value per problem instance over the 25 replications. This indicates that the model consistently generates good solutions. Therefore, we select this configuration for the remaining experiments.

5.4 Results on the impact of the model

This section provides the results on the impact of the model on the current situation, as described in the experimental design in Section 5.2.2. Section 5.4.1 provides the simulated results of (Exp3 – Exp5) on the impact of the model compared to the current situation. Section 5.4.2 provides the results of Exp6 on the impact of a simple version of the model in practice after implementation. Regarding the results, the timestamps have the format “d:hh:mm:ss”.

5.4.1 Simulated results on the impact of the model

This section provides the simulated results of (Exp3 – Exp5) on the impact of the model compared to the current situation. The symbols (🡑) and (🡓) indicate a significant increase or decrease of the KPI with an alpha of 0.01, respectively. The absence of these symbols indicates no significant difference. The colors green and red represent a better and worse performance, respectively.

Experiment 3 | Evaluating the effect of optimizing the schedules of the stages simultaneously

Recall that Exp3 evaluates the effect of (i) optimizing the schedules of the stages simultaneously compared to (ii) the current situation, which optimizes the stages separately. We refer to the former as “Simultaneous optimization” and to the latter as “Current situation”. For both scenarios, the production routes of the jobs remain at the default route. Table 19 provides the experimental results of the KPIs per demand level for both scenarios. The standard deviation of the KPIs is reported between brackets. The detailed results per problem instance are in Appendix 17.

Table 19 | Results of the comparison between scheduling the stages separately and simultaneously

Demand	Scenario	C_{max}	T_{tot}	CT_{tot}	BT_{avg}	IBC_{max}	Feasible
Low	Current situation	4:10:56:12 (10:35:18)	7:40:20 (15:37:45)	4:12:28:00 (10:12:04)	2:04:51 (37:24)	40.7 (5.0)	100%
	Simultaneous optimization	3:23:26:03 (8:40:25)	0	3:08:31:42 (9:40:19)	3:10:18 (33:16)	48.9 (4.3)	100%
Normal	Current situation	6:08:26:18 (6:45:32)	21:22:56 (1:08:13:46)	6:11:30:30 (7:06:07)	0:04:53:58 (1:44:03)	54.5 (8.7)	74%
	Simultaneous optimization	6:07:32:24 (7:37:12)	0	5:17:09:24 (10:52:21)	5:15:34 (48:45)	57.4 (5.4)	70%
High	Current situation	8:04:57:42 (2:40:03)	4:12:49:31 (5:04:59:48)	8:03:52:30 (6:01:44)	8:07:48 (1:11:02)	70.1 (6.5)	8%
	Simultaneous optimization	8:08:41:13 (5:07:42)	0:01:51:24 (7:32:08)	8:01:13:24 (8:08:25)	7:09:28 (56:56)	66.0 (6.4)	22%

Furthermore, Table 20 provides the average performance difference between optimizing the stages simultaneously compared to optimizing the stages separately.

Table 20 | Performance difference between optimizing stages simultaneously compared to separately

Demand	C_{max}	T_{tot}	CT_{tot}	BT_{avg}	IBC_{max}	Feasibility
Low	↓ -10.8%	↓ -100.0%	↓ -25.8%	↑ 52.4%	↑ 20.2%	0.0%
Normal	-0.6%	↓ -100.0%	↓ -11.8%	7.3%	5.2%	-4.0%
High	↑ 1.9%	↓ -98.3%	-1.4%	-12.0%	↓ -5.8%	↑ 14.0%
Average difference	**-3.2%**	↓ **-99.4%**	**-13.0%**	**15.9%**	**6.5%**	**3.3%**

Based on the results in Table 19 and Table 20, we observe that simultaneous optimization of the stages results in a significant performance increase for the T_{tot} and CT_{tot} compared to the current situation. Moreover, for the low demand instances, the C_{max} and CT_{tot} significantly improve at the expense of the BT_{avg} and IBC_{max}, whereas this is the other way around for the high demand instances. This is logical since for the low demand scenarios there are fewer IBCs needed than for the high demand scenarios. Therefore, it is possible to let the IBCs wait longer in the buffers (i.e., increasing BT_{avg}) to be able to create a more efficient sequence of products on the machines in the next production stage, which results in a decrease in the C_{max} and CT_{tot}. This is the other way around for the high demand scenarios; the high demand scenarios need to decrease the IBC_{max} to increase the feasibility. This results in less efficient sequences of jobs on the machines, resulting in a slightly higher C_{max}.

Furthermore, Table 21 provides an overview of the scheduled cleaning time per stage for both scenarios. We observe that the current situation has very little cleaning time at the mixing stage, whereas the IBC-filling stage has a lot of cleaning time. This is logical since the current situation first optimizes the mixing schedule based on the CT_{tot} and T_{tot}. After that, ESA creates the IBC-filling schedule by minimizing the flowtime FT_{tot}, without changing the mixing schedule.

Table 21 | Cleaning time per production stage

Scenario	IBC-filling	Mixing	Discharging	Total
Current situation	2:20:37:00 (19:26:49)	2:00:34:06 (9:30:00)	1:12:05:54 (9:30:58)	6:09:17:00 (1:12:42:13)
Simultaneous optimization	1:13:32:40 (19:41:48)	2:21:24:00 (21:13:44)	1:06:01:30 (11:12:53)	5:16:58:10 (1:23:08:30)

Experiment 4 | Evaluating the effect of changing the production routes

Recall that Exp4 evaluates the effect of the ability to allow (i) only the default production route and (ii) changing the production route to any other eligible production route. We refer to the first scenario as "Model (default routes)" and to the second scenario as "Model (eligible routes)". Table 22 provides the experimental results of the KPIs for both scenarios, of which the standard deviation is reported between brackets. The detailed results per problem instance are in Appendix 18.

Table 22 | Results of the comparison between allowing the default routes and all eligible routes

Demand	Scenario	C_{max}	T_{tot}	CT_{tot}	BT_{avg}	IBC_{max}	Feasible
Low	Model (default route)	3:23:26:03 (8:40:25)	0	3:08:31:42 (9:40:19)	3:10:18 (33:16)	48.9 (4.3)	100%
	Model (eligible routes)	3:17:38:42 (5:01:26)	0	3:13:22:54 (11:21:46)	2:07:48 (24:21)	42.2 (4.0)	100%
Normal	Model (default route)	6:07:32:24 (7:37:12)	0	5:17:09:24 (10:52:21)	5:15:34 (48:45)	57.4 (5.4)	70%
	Model (eligible routes)	5:00:49:40 (4:53:28)	0	6:01:58:54 (17:51:21)	3:35:56 (45:00)	49.4 (4.9)	98%
High	Model (default route)	8:08:41:13 (5:07:42)	1:51:24 (7:32:08)	8:01:13:24 (8:08:25)	7:09:28 (56:56)	66.0 (6.4)	22%
	Model (eligible routes)	6:11:12:51 (3:41:53)	0	9:07:26:36 (7:39:36)	4:14:52 (32:32)	52.8 (5.2)	92%

Table 23 | Average performance difference between allowing eligible routes instead of default routes

Demand	C_{max}	T_{tot}	CT_{tot}	BT_{avg}	IBC_{max}	Feasibility
Low	↓ -6.1%	0.0%	6.0%	↓ -32.8%	↓ -13.7%	0.0%
Normal	↓ -20.3%	0.0%	6.4%	↓ -31.6%	↓ -14.0%	↑ 28.0%
High	↓ -22.7%	↓ -100.0%	↑ 15.6%	↓ -40.7%	↓ -20.0%	↑ 70.0%
Average difference	**↓ -16.3%**	**↓ -33.3%**	**↑ 9.4%**	**↓ -35.0%**	**↓ -15.9%**	**↑ 32.7%**

Furthermore, Table 23 provides the average performance difference when considering the eligible routes compared to only considering the default routes. Based on the results in Table 22 and Table 23, we observe that allowing changing the production routes results in a significant performance improvement for the C_{max}, T_{tot}, BT_{avg}, and IBC_{max} for all demand levels. Moreover, the feasibility increases. However, the CT_{tot} increases significantly as well for the high demand instances and on average over all instances. This is logical since allowing more changeovers results in more flexibility to reduce the C_{max}, T_{tot}, BT_{avg}, and IBC_{max}, and also increase the feasibility. However, having more changeovers may result in a higher CT_{tot}. Nevertheless, the improvement on the other KPIs outweighs the increase of the CT_{tot}.

Table 24 provides an overview of the number of operations and the total processing- and cleaning time per machine for both scenarios.

Table 24 | The impact of considering eligible routes instead of default routes

Stage / Machine		Number of operations		Processing and cleaning time	
		Model (default route)	Model (eligible routes)	Model (default route)	Model (eligible routes)
Mixing	Z401 (200L)	32.2 (11.5)	34.0 (12.3)	2:01:56:20 (0:17:10:39)	2:04:58:16 (0:18:39:49)
	Z402 (1.5K)	23.8 (7.4)	↑ 40.2 (12.8)	2:08:47:48 (0:18:44:21)	↑ 4:02:15:42 (1:09:45:22)
	Z403 (3.0K)	42.5 (11.0)	38.7 (11.8)	3:21:48:56 (1:02:00:08)	3:14:34:06 (1:03:01:17)
	Z404 (4.5K)	38.2 (11.2)	38.6 (10.6)	4:18:51:46 (1:09:42:47)	4:10:07:10 (1:05:45:02)
	Z405 (3.0K)	40.0 (12.0)	38.3 (11.4)	3:17:36:00 (1:01:56:23)	3:14:46:02 (1:02:26:04)
	Z407 (10K)	23.3 (3.4)	↑ 46.0 (10.1)	3:08:11:00 (0:10:46:40)	↑ 4:21:13:54 (1:02:52:34)
	Z408 (10K)	59.7 (12.4)	↓ 45.8 (10.3)	5:00:30:30 (1:02:04:06)	4:21:18:06 (1:03:17:14)
	Z409 (Tumbler)	43.3 (22.1)	↓ 21.4 (11.7)	1:08:30:00 (0:16:34:13)	↓ 0:16:01:30 (0:08:48:06)
Discharge	Z410 (Votech)	79.5 (27.8)	↓ 41.9 (13.0)	5:20:20:30 (2:00:15:20)	↓ 3:10:30:32 (1:03:56:21)
	Z412 (BTH)	8.7 (2.7)	↑ 27.8 (8.4)	1:07:39:18 (0:10:53:29)	↑ 4:03:07:06 (1:07:50:29)
	Z420 (Dinnissen)	34.5 (17.2)	31.1 (15.0)	1:18:55:12 (0:21:43:56)	↓ 1:08:44:18 (0:14:21:12)

From Table 24, we observe that the model that allows changing the production routes allocates jobs from the route Z409 – Z410 to the Z402. This is logical since the Z410 seems to be the bottleneck when looking at the total processing- and cleaning time. Moreover, the Z402 has its discharging line. Therefore, the Z402 route does not require to buffer IBCs between the mixing and discharging stage, resulting in fewer IBCs needed, which enhances satisfying the IBC-capacity constraint. Furthermore, the model that allows changing the production routes allocates jobs from the Z408 mixer to the Z407 mixer.

Experiment 5 | Comparing the performance of the current situation to the performance of the model
This section compares the performance of the current situation to the performance of the model. Table 25 provides an overview of the results per instance for both scenarios. The standard deviation of the KPIs is reported between brackets. For every instance and scenario, this experiment performs 25 replications. Furthermore, Table 26 provides the average performance difference between the selected model and the current situation for each problem instance.

Table 25 | Results of the comparison between the current situation and the proposed model

Instance	Scenario	C_{max}	T_{tot}	CT_{tot}	BT_{avg}	IBC_{max}	Feasible
1	Current situation	4:19:28:09 (5:50:28)	0	4:21:23:48 (5:28:53)	1:45:41 (32:11)	38.9 (4.5)	100%
	Proposed model	3:22:08:00 (2:28:58)	0	3:23:49:00 (4:21:03)	2:09:02 (24:04)	41.2 (3.9)	100%
2	Current situation	4:02:24:14 (6:33:56)	15:20:40 (19:23:35)	4:03:32:12 (4:02:06)	2:24:59 (31:51)	42.5 (4.9)	100%
	Proposed model	3:13:09:24 (1:50:37)	0	3:02:56:48 (4:13:17)	2:06:30 (25:04)	43.3 (3.9)	100%
3	Current situation	6:14:34:19 (2:16:29)	1:17:43:57 (1:11:05:27)	6:17:29:12 (3:44:30)	6:17:28 (1:13:58)	59.7 (8.4)	48%
	Proposed model	5:05:05:48 (2:24:30)	0	6:18:23:24 (6:40:45)	3:59:05 (45:07)	49.7 (4.8)	100%
4	Current situation	6:02:18:16 (3:07:05)	1:01:55 (5:09:36)	6:05:31:48 (3:48:40)	3:29:39 (44:22)	49.4 (5.3)	100%
	Proposed model	4:20:33:33 (2:15:14)	0	5:09:34:24 (6:44:27)	3:12:33 (31:11)	49.0 (5.1)	96%

Instance	Scenario	C_{max}	T_{tot}	CT_{tot}	BT_{avg}	IBC_{max}	Feasible
5	Current situation	8:03:59:12 (2:44:52)	3:55:00 (10:51:49)	8:03:59:24 (6:17:01)	7:55:15 (1:01:03)	71.0 (6.5)	4%
	Proposed model	6:08:44:38 (2:50:49)	0	9:07:49:48 (8:59:09)	4:11:00 (33:01)	52.1 (5.4)	92%
6	Current situation	8:05:56:12 (2:14:07)	8:21:44:02 (3:22:05:17)	8:03:45:36 (5:53:26)	8:20:20 (1:19:03)	69.2 (6.4)	12%
	Proposed model	6:13:41:04 (2:39:55)	0	9:07:03:24 (6:13:28)	4:18:43 (32:15)	53.6 (5.1)	92%

Table 26 | Average performance difference between the proposed model and the current situation

Demand	Instance	C_{max}	T_{tot}	CT_{tot}	BT_{avg}	IBC_{max}	Feasibility
Low	1	↓ -18.5%	0.0%	↓ -18.4%	22.1%	5.8%	0.0%
	2	↓ -13.5%	↓ -100.0%	↓ -24.7%	-12.8%	1.9%	0.0%
Normal	3	↓ -21.1%	↓ -100.0%	-0.6%	↓ -36.7%	↓ -16.8%	↑ 52.0%
	4	↓ -20.3%	↓ -100.0%	↓ -13.3%	-8.2%	-0.6%	-4.0%
High	5	↓ -22.1%	↓ -100.0%	↑ 14.2%	↓ -47.2%	↓ -26.6%	↑ 88.0%
	6	↓ -20.3%	↓ -100.0%	↑ 13.9%	↓ -48.3%	↓ -22.6%	↑ 80.0%
Average difference		**↓ -19.9%**	**↓ -100.0%**	**-1.1%**	**↓ -37.6%**	**↓ -12.7%**	**↑ 36.0%**

Based on the results in Table 25 and Table 26, we observe a significant performance increase for all problem instances for the C_{max} and T_{tot}. Moreover, the feasibility increases significantly for most instances. However, the performance of the CT_{tot} decreases significantly for the high demand instances. This is logical since the performance on the BT_{avg} and IBC_{max} increases for these high demand instances to enhance finding feasible solutions. Therefore, this results in more cleaning time. Nevertheless, we still observe an average performance increase on all KPIs. All in all, the weekly production throughput, which is the main KPI of Euroma, increases from 300 jobs for the current situation to 400 jobs per week for the proposed model, which meets the desired level of Euroma.

5.4.2 Results of the implementation in practice

This section provides the results of Exp6, which evaluates the performance of a simple version of the model after implementation in practice.

Experiment 6 | Evaluate the performance of the simple version of the model in practice

Recall that this experiment evaluates the performance in practice of the model that only optimizes the mixing schedule compared to the current situation. In Figure 31, the red- and blue lines indicate the scheduled- and realized average cleaning time per job per year-week, respectively. These cleaning times are obtained from the contamination matrix. Note that we started creating the production schedules in week 3 of 2021. The difference between the scheduled and realized cleaning times occurred due to priority jobs that were not scheduled that caused extra cleaning time. The scheduled T_{tot} was zero for all instances.

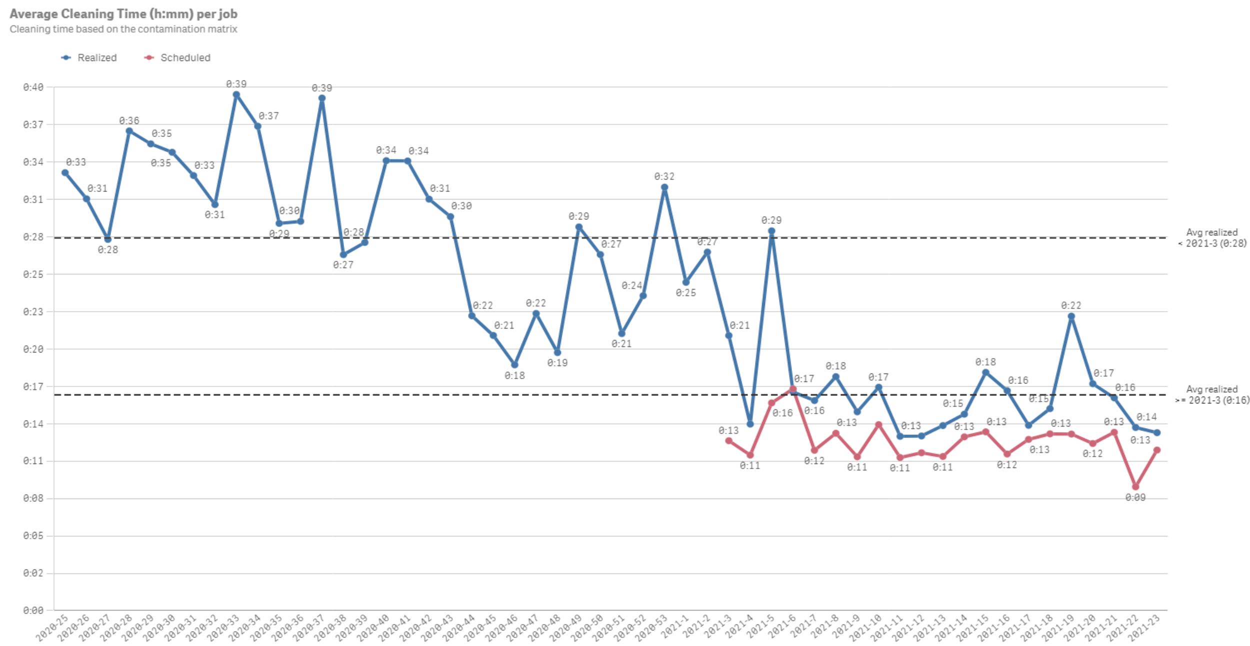


Figure 31 | Comparison between the scheduled and realized average cleaning time per job over time

Furthermore, Figure 32 provides boxplots of the average realized cleaning time per job per year-week before- and after implementation of the model in practice. To test if there is a significant reduction in the cleaning time after implementation, we perform a two-sample t-test in which we assume unequal variances. Moreover, we assume that all samples are independent identically distributed. This is reasonable to assume since there seems to be no dependence between the weeks, since Euroma schedules a new set of jobs every week. We set alpha at 0.005, resulting in a p-value of 5.2E-14. Therefore, we reject the null hypotheses of equal average cleaning times per job before- and after implementation. Figure 33 provides the corresponding 99%–CI, of which the statistical results are in Appendix 19. Following from the 99%–CI, the total cleaning time is reduced between 26.3% and 49.1%. This results in a weekly cleaning time saving of the mixers between [0:13:21:14 – 1:00:55:50].

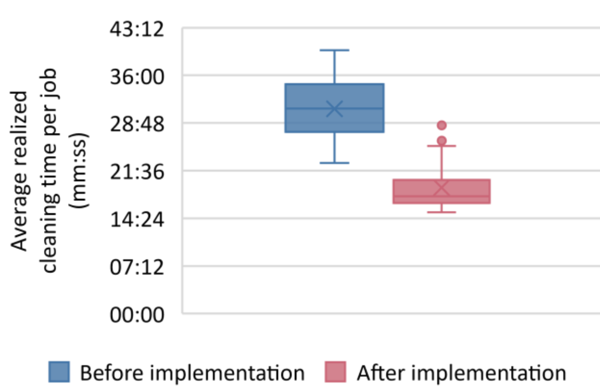


Figure 32 | Average realized cleaning time per job before- and after implementation

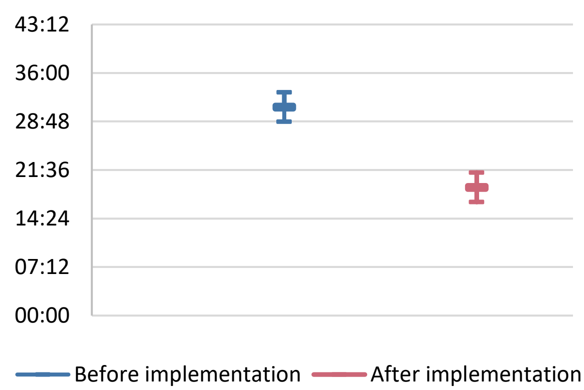


Figure 33 | 99%-CI of the average cleaning time per job before- and after implementation

5.5 Summary of the experiments

The main goal of this chapter is to answer the fourth research question: Which alternative model performs best compared to the current situation under different experimental settings? To answer this question, Section 5.1 provides a summary of six problem instances that are extracted from the company data that we use for the experiments. These problem instances consider low, normal, and high demand.

Section 5.2 first describes the experimental design to identify a suitable weight set for the objective function and the most promising model configuration among the possible configurations (i.e., the construction- and improvement heuristics including the corresponding parameter sets that result in the best solutions according to the objective function). This section also describes the experimental design to evaluate the effect of (i) optimizing the schedules of the production stages simultaneously instead of separately, (ii) considering all eligible production routes instead of the default production routes, and (iii) the performance of the proposed model compared to the current situation. Finally, this section describes the experimental design to evaluate the performance in practice of a model that only optimizes the mixing schedule compared to the current situation.

Section 5.3 provides the experimental results of the objective weight tuning and the different model configurations. Based on these results, we select the objective weight set $\{14\%, 14\%, 14\%, 28\%, 30\%\}$, which corresponds with the weights $\{w_{C_{max}}, w_{T_{tot}}, w_{CT_{tot}}, w_{FT_{tot}}, w_{IBC_{penalty}}\}$, respectively. Besides that, the results indicate that the model configuration with the NEH construction heuristic, simulated annealing improvement heuristic, and a random neighborhood structure results in the best objective values while still obtaining high feasibility. Therefore, we select this configuration for the remaining experiments.

Section 5.4 first provides the experimental results on the effect of optimizing the schedules of the production stages simultaneously instead of separately. These results show that simultaneously optimizing the stages results in a significant performance increase for the T_{tot} and CT_{tot} compared to the current situation. Moreover, for the low demand instances, the C_{max} significantly improves at the expense of the BT_{avg} and IBC_{max}, whereas this is the other way around for the high demand instances. Moreover, Section 5.4 provides the experimental results on the effect of considering all eligible production routes instead of the default production routes. These results show that allowing changing the production routes results in a significant performance improvement for the C_{max}, T_{tot}, BT_{avg}, and IBC_{max} for all demand levels. Moreover, the feasibility increases. However, the CT_{tot} increases significantly as well for the high-demand instances and on average over all instances

Furthermore, Section 5.4 compares the performance of the proposed model with the current situation. The results show that the proposed model significantly reduces the C_{max} and T_{tot} for all problem instances compared to the current situation. Moreover, the feasibility increases significantly for most instances. However, the CT_{tot} increases significantly for the high demand instances. This is logical since the BT_{avg} and IBC_{max} decrease significantly for these high demand instances to enhance finding feasible solutions, which results in more cleaning time. All in all, the weekly production throughput, which is the main KPI of Euroma, increases from 300 jobs for the current situation to 400 jobs per week for the proposed model, which meets the desired level of Euroma.

Finally, Section 5.4 evaluates the performance in practice of the model that only optimizes the mixing schedule compared to the current situation. The results show that the total cleaning time was reduced between 26.3% and 49.1% after implementing the model in practice. This results in a weekly cleaning time saving of the mixers between [0:13:21:14 – 1:00:55:50].

6. Model implementation

This chapter describes how the model that we propose can be implemented in practice. Section 6.1 explains the implementation architecture of the model. Section 6.2 describes the output of the model and then explains the dashboard that visualizes the data. Finally, Section 6.3 elaborates upon the techniques to verify the input of the model, the model itself, and the output of the model.

6.1 Implementation architecture

This section explains the implementation architecture of the model by using the data flow in Figure 34.

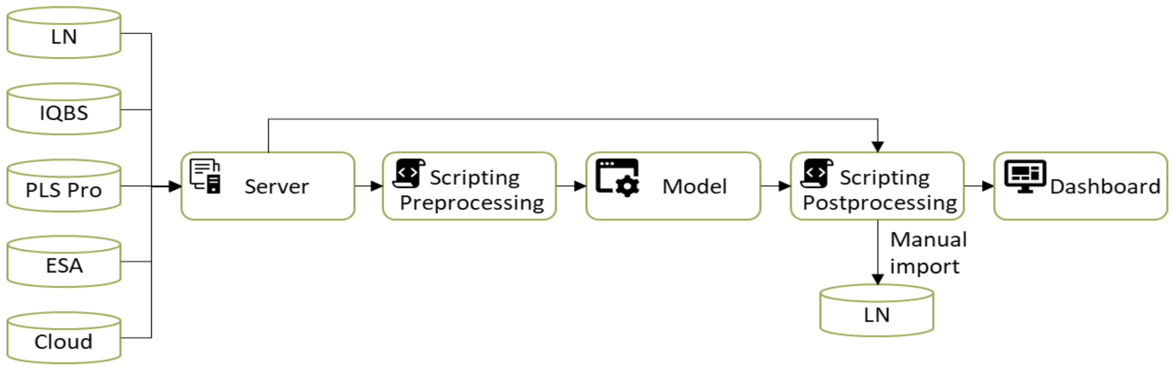


Figure 34 | Data flow of model implementation

The model needs data from five databases. The IT systems of these sources (e.g., LN, ESA, or IQBS) and the data that each source manages are described in Section 2.4. The server stores the extracted data in dedicated folders. The SQL scripts preprocess these extracted files into input data tables for the model. The SQL scripts apply techniques for creating, e.g., compact memory representations of complex keys. For example, an order number of Euroma consists of a string of nine characters and the scripts transform these strings into job numbers consisting of unique integer values. This way, large tables such as the contamination matrix are more efficient in memory usage as integers use fewer bytes than characters. Moreover, integers have faster CPU times than characters (Aliyu & Zirra, 2013).

We use the programming language Delphi to implement the model. Delphi allows compiling the code into an executable application that allows communication with scripts, databases, and the Windows API. The SQL scripts post-process the output data of the model to (i) a file containing the schedule that can be imported manually into the ERP system LN, and (ii) a set of files that a dashboard visualizes for the user. Regarding the latter, Section 6.2 explains the output of the model and the dashboard.

6.2 Model output & dashboard

This section first explains the output of the model and then explains the dashboard that visualizes the data. The output data of the model consists of two tables: (i) the production schedule (see Table 27) and (ii) the IBC status of the orders at every minute (see Table 28).

Table 27 | Production schedule model output

Machine	Nr	Task	OrderNr	Article	Route	Start	Stop
IBC Filling 1	1	IBC Fill	J00019978	59883	13	5-3-21 0:15	5-3-21 0:55
IBC Filling 1	2	Cleaning	-	-	-	5-3-21 0:55	5-3-21 1:05
IBC Filling 1	3	IBC Fill	J00017891	56832	16	5-3-21 1:05	5-3-21 3:05
IBC Filling 1	4	Cleaning	-	-	-	5-3-21 3:05	5-3-21 3:35
IBC Filling 1	5	IBC Fill	J00020210	64739	7	5-3-21 3:35	5-3-21 4:55
IBC Filling 1	6	IBC Fill	J00019181	62260	15	5-3-21 4:55	5-3-21 6:15
IBC Filling 1	7	IBC Fill	J00019433	35672	12	5-3-21 6:15	5-3-21 8:55
IBC Filling 1	8	IBC Fill	J00019047	57098	4	5-3-21 8:55	5-3-21 10:15
IBC Filling 1	9	IBC Fill	J00019980	59883	8	5-3-21 10:15	5-3-21 10:55

Table 28 | IBC pool status model output

OrderNr	Timestamp	IBC Status	IBCs
-	5-4-21 13:05	Cleaning	6
J00009243	5-4-21 13:05	Production	2
J00017499	5-4-21 13:05	Production	1
J00018145	5-4-21 13:05	Production	4
J00018620	5-4-21 13:05	Production	2
J00019563	5-4-21 13:05	Production	5
J00020097	5-4-21 13:05	Production	3
J00020156	5-4-21 13:05	Production	2
J00020160	5-4-21 13:05	Production	6

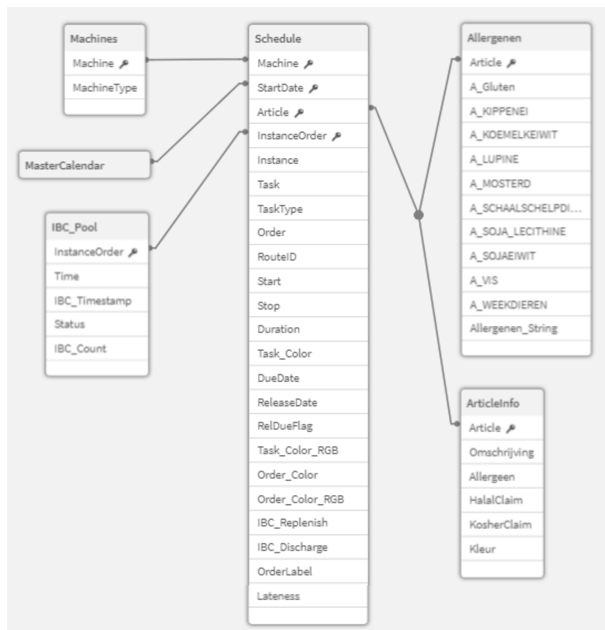
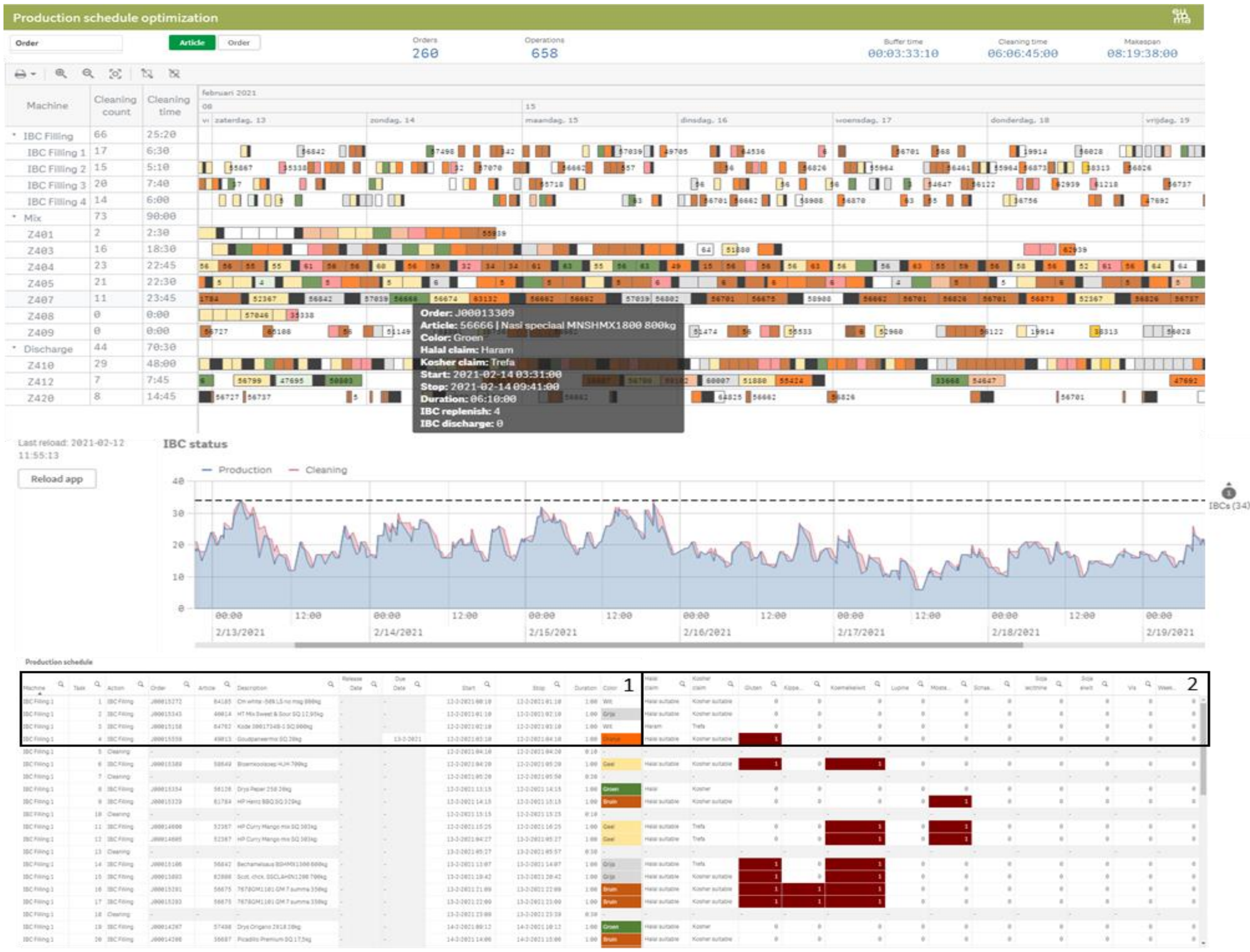


Figure 35 | Dashboard data model

The output of the model gets loaded into a data model that also consists of article information tables with, e.g., the article descriptions, article colors, claims, and allergens. Figure 35 provides the data model of the dashboards.

The data model gets visualized by a dashboard (see Figure 36). The top of the dashboard visualizes the key performance indicators (KPIs), e.g., the makespan, total cleaning time, total tardiness, and the total buffer time. The Gantt chart visualizes the schedule. Each row in the Gantt chart shows the sequence of jobs on a machine including important KPIs (e.g., the number of cleanings and the total cleaning time of that machine). The blocks in the sequence represent operations and the color of a block corresponds with the color of the article of the operation, whereas the black blocks indicate cleanings. The label on a block corresponds with the article number. Moreover, a pop-up with detailed information of the operation (e.g., the order number, color, claims, start- and finish times, and the IBC information) appears when hovering over a block.

The stacked line chart shows the status of the IBCs over time, where the blue area indicates the number of IBCs in production and the red area indicates the number of IBCs in cleaning, as described in Appendix 5. The dashed line in the line chart indicates the number of IBCs needed. The stacked line chart aligns with the Gantt chart regarding the time dimension.



Machine	Task	Action	Order	Article	Description	Release Date	Due Date	Start	Stop	Duration	Color
IBC Filling 1	1	IBC Filling	J00015272	64185	Cm white -50% LS no msg 800kg	-	-	12-2-2021 00:10	12-2-2021 01:10	1:00	Wit
IBC Filling 1	2	IBC Filling	J00015343	40014	HT Mix Sweet & Sour SQ 12,95kg	-	-	12-2-2021 01:10	12-2-2021 02:10	1:00	Grijs
IBC Filling 1	3	IBC Filling	J00015158	64762	Kode 3001734B-1 SQ 900kg	-	-	12-2-2021 02:10	12-2-2021 03:10	1:00	Wit
IBC Filling 1	4	IBC Filling	J00015559	49013	Goudpaneermix SQ 20kg	-	13-2-2021	12-2-2021 03:10	12-2-2021 04:10	1:00	Oranje

Halal claim	Kosher claim	Gluten	Kippe...	Koemelkeiwit	Lupine	Moste...	Schaa...	Soja lecithine	Soja eiwit	Vis	Week...
Halal suitable	Kosher suitable	0	0	0	0	0	0	0	0	0	0
Halal suitable	Kosher suitable	0	0	0	0	0	0	0	0	0	0
Haram	Trefa	0	0	0	0	0	0	0	0	0	0
Halal suitable	Kosher suitable	1	0	0	0	0	0	0	0	0	0

Figure 36 | Dashboards of the model

The table in Figure 36 (marked with the numbers 1 and 2) provides detailed information regarding the production schedule. Each row in the table corresponds with a task, which is either processing an operation (e.g., IBC-filling, mixing, or packaging), cleaning, or a maintenance activity. The table provides the order information of the operations, including optional release- and due dates. For the latter, the table highlights the due dates in red that are not met. The table also highlights the colors and allergens of the articles.

6.3 Model verification

This section elaborates upon the techniques to verify the input of the model, the model itself, and the output of the model. These verification methods are necessary to enhance the model's credibility and to ensure food safety by preventing incorrect schedules that might lead to cross-contamination of allergens.

At first, several procedures verify the input data. For instance, these procedures verify whether the input has the right data format and whether the input contains null values. Moreover, these procedures verify whether the colors in the input data match with the colors in the contamination matrix. Besides that, procedures check whether the number of IBCs that discharge the last operation of a job equals zero, otherwise, there might be a mistake in the production route or the IBC-usage calculation. Regarding the latter, the procedures also verify whether the number of IBCs that replenish the Tumbler mixer (Z409) is equal to one, as this mixer can only rotate one IBC at a time, as described in Section 2.1.2. Once the verification procedures flag an error, the scripts stop and the error is logged in a file.

The model consists of small procedures that each have a clear purpose. Each procedure is extensively tested by varying the input values and verifying whether the output is as expected. Besides that, the output of the model is extensively verified. Separate procedures verify whether the scheduling constraints are met for the output solution, e.g., whether jobs with release times start after the release time, or if there are no activities during production-stops of machines. Moreover, separate procedures verify specific food safety constraints such as the claim constraints and whether there are cleanings scheduled that correspond with the contamination matrix. Appendix 11 provides a pseudo-code of the verification procedure that checks whether the claim constraints are satisfied.

Finally, experts (e.g., the planners, production manager, food quality officers, and control room operators) extensively verify and review the output of the model by using the dashboard (see Section 6.2).

7. Conclusion & recommendations

In this chapter, Section 7.1 summarizes the main findings and answers the main research question. Subsequently, Section 7.2 provides recommendations to Euroma. Section 7.3 discusses the limitations of this research and provides direction for future research. Finally, Section 7.4 describes the contribution of this research to science.

7.1 Conclusion

The facility of Euroma in Zwolle is at its limits as almost every square meter is occupied. Still, this facility cannot satisfy customer demand. An analysis showed that the mixers need cleaning 15% of the time and that they are idle 30% of the time due to waiting on shared resources (e.g., IBCs and operators). As a result, the production throughput is less than 300 mixtures per week, whereas 400 mixtures per week are required to satisfy the demand. Therefore, the research question is:

"How to optimize the multi-stage production schedule to achieve the desired throughput?"

The scheduling problem comprises several unique criteria, e.g., restricted sequences of jobs on machines and shared resources over multiple production stages. For this problem, we proposed 12 model configurations that each consists of (i) a construction heuristic, (ii) an improvement heuristic, and (iii) a neighborhood structure. We extended the construction heuristics such that they always generate feasible solutions regarding the claim constraints. Moreover, we proposed a decoding- and a corrective backtracking algorithm to determine the start- and finish times of the jobs and cleanings.

We performed experiments to tune the weights in the objective function and to find the most promising model configuration according to this objective function. The results indicate that the configuration with the extended NEH construction heuristic, the simulated annealing improvement heuristic, and a random neighborhood structure results in the best objective values. We selected this configuration for the remaining experiments. Subsequently, we evaluated the following scenarios:

1. Optimize the schedules of the production stages simultaneously instead of separately;
2. Consider all eligible production routes instead of only the default production routes;
3. Optimize the schedules of the production stages simultaneously and consider all eligible production routes instead of optimizing the schedules of the stages separately and only considering the default production routes.

Table 1 provides the average difference per KPI of these three scenarios. The symbols (↑) and (↓) indicate a significant increase or decrease of the KPI with an alpha of 0.01, respectively. The absence of these symbols indicates no significant difference. The colors green and red represent a better and worse performance, respectively.

Table 1 | Average performance difference per scenario

Scenario	Makespan	Tardiness	Cleaning time	Buffer time	IBCs needed	Feasibility
(1) Simultaneous versus separate optimization of production stages	-3.2%	↓ -99.4%	-13.0%	15.9%	6.5%	3.3%
(2) Allowing eligible routes versus only allowing default routes	↓ -16.3%	↓ -33.3%	↑ 9.4%	↓ -35.0%	↓ -15.9%	↑ 32.7%
(3) Proposed model versus current situation	↓ -19.9%	↓ -100.0%	-1.1%	↓ -37.6%	↓ -12.7%	↑ 36.0%

We conclude that optimizing the production stages simultaneously instead of separately significantly improves the total tardiness. Allowing the model to allocate an eligible production route to a job instead of only considering the default jobs, results in a significant performance increase for all KPIs, except for the cleaning time; the performance of the cleaning time decreases significantly. This is reasonable since allowing more changeovers results in more flexibility to improve the other KPIs. Improving the other KPIs outweighs the increase in the cleaning time.

Table 29 provides the 99%–CI of the KPIs to compare the performance of the current situation with the performance of the proposed model per demand level. The timestamps have the format “d:hh:mm:ss”.

Table 29 | 99%–CI of the current situation and the proposed model

Demand	Model	Current situation	Proposed model	Difference
Low (± 200 jobs)	Makespan	[4:07:04:46 - 4:14:47:38]	[3:15:48:53 - 3:19:28:31]	↓ -16.2%
	Tardiness	[0:01:58:44 - 0:13:21:57]	0:00:00:00	↓ -100.0%
	Cleaning time	[4:08:45:02 - 4:16:10:58]	[3:09:14:33 - 3:17:31:15]	↓ -21.3%
	Buffer time	[0:01:51:14 - 0:02:18:29]	[0:01:58:56 - 0:02:16:41]	2.4%
	IBCs needed	[38.9 – 42.5]	[40.8 – 43.7]	3.7%
	Feasibility	100%	100%	0.0%
Normal (± 300 jobs)	Makespan	[6:05:58:34 - 6:10:54:02]	[4:23:02:47 - 5:02:36:35]	↓ -20.7%
	Tardiness	[0:09:38:30 - 1:09:07:22]	0:00:00:00	↓ -100.0%
	Cleaning time	[6:08:55:16 - 6:14:05:44]	[5:19:28:38 - 6:08:29:10]	↓ -6.1%
	Buffer time	[0:04:16:04 - 0:05:31:53]	[0:03:19:33 - 0:03:52:20]	↓ -26.5%
	IBCs needed	[51.4 – 57.7]	[47.6 – 51.1]	↓ -9.5%
	Feasibility	74%	98%	↑ 24.0%
High (± 400 jobs)	Makespan	[8:03:59:24 - 8:05:56:00]	[6:09:52:02 - 6:12:33:42]	↓ -21.2%
	Tardiness	[2:15:17:31 - 6:10:21:32]	0:00:00:00	↓ -100.0%
	Cleaning time	[8:01:40:44 - 8:06:04:16]	[9:04:39:10 - 9:10:14:02]	↑ 14.1%
	Buffer time	[0:07:41:56 - 0:08:33:41]	[0:04:03:01 - 0:04:26:44]	↓ -47.8%
	IBCs needed	[67.8 – 72.5]	[50.9 – 54.7]	↓ -24.7%
	Feasibility	8%	92%	↑ 84.0%

We conclude that, compared to the current situation, the proposed model significantly improves almost every KPI on every demand level, except for the cleaning time at the high demand level; the cleaning time increases significantly. This is reasonable since allowing more changeovers results in more flexibility to reduce the makespan, tardiness, buffer time, and the number of IBCs needed. However, having more changeovers may result in a higher cleaning time. Nevertheless, the improvements on the other KPIs outweighs the increase in the cleaning time. All in all, the weekly production throughput, which is the main KPI of Euroma, increases from 300 jobs for the current situation to 400 jobs per week for the proposed model, which meets the desired level of Euroma.

Finally, we implemented a model in practice to optimize the mixing schedules by minimizing the total cleaning time and total tardiness. Based on the results, we conclude from the 99%–CIs that the cleaning time reduction is between [26.3% – 49.1%] compared to the situation before implementing the model. This results in a weekly cleaning time saving of the mixers between [0:13:21:14 – 1:00:55:50].

7.2 Recommendations

Based on our evaluation of the proposed models, we recommend implementing the scheduling model that can optimize the schedules of the production stages simultaneously and that can allocate production routes to jobs. This model outperforms all other models and results in significantly better performance than the current scheduling procedures. Moreover, the proposed scheduling model can cope with different demand scenarios. To configure the model, we recommend using the extended NEH construction heuristic and the simulated annealing improvement heuristic with a random neighborhood structure, as this configuration outperforms the other configurations.

Furthermore, to enhance the quality of the solutions of the model, we recommend enriching the input data by logging the processing times of the key production steps (e.g., the filling-, discharging- and liquid dosing steps). Moreover, we recommend monitoring the cleaning times to improve the accuracy of the contamination matrix.

Our research focused on evaluating the performance of the proposed models. Chapter 6 describes how we implemented a simple version of the model in practice. However, we did not study the costs and consequences for stakeholders and IT systems for the implementation of the proposed model. Therefore, we recommend investigating these costs and consequences before creating a plan to implement the proposed model.

7.3 Limitations & future research

This section discusses the limitations of this research and provides directions for future research to improve the production performance and to enhance our research.

First, we limited the research scope to the IBC-filling-, mixing-, and discharging processes. We did not consider processes outside the scope of this research (e.g., filling the silos or palletizing the bags), as there was no incentive to assume that these are bottlenecks. Nevertheless, implementing the proposed model might shift the bottleneck to a process outside the scope of this research. In that case, the focus should be on the new bottleneck process to improve the overall production performance.

Second, the model might not be able to obtain a feasible solution for some problem instances, as the limited IBC-capacity is modeled as a soft-constraint. Therefore, future research can focus on a corrective algorithm that always obtains feasible solutions regarding the IBC-capacity constraint. As described in Section 4.4, such an algorithm must be able to change the sequence of jobs on machines and determine the start- and finish times of the operations while still satisfying all remaining constraints.

Third, the initial solution seems to have no influence on the final objective value for the improvement heuristics SA and SATL, as described in Section 5.3.2. This indicates that SA and SATL both have a good improvement ability, regardless of the initial solution. Therefore, to improve the model even further, we suggest focusing on the improvement heuristics.

Finally, the performance of the model is not evaluated in a stochastic environment with, e.g., machine breakdowns. Therefore, future research should focus on evaluating the performance of the model under stochastic circumstances by using a simulation study. This allows testing strategies to make the model solutions more robust. For instance, this can be achieved by scheduling more buffer time for operations before they need processing on the bottleneck machine. This way, if the machine before the bottleneck machine fails, there still is a buffer to keep the bottleneck machine operational.

7.4 Contribution of this research

This section highlights the contribution of our research to the scientific body of knowledge.

First, following from Section 3.3, HFS problems in the literature each cover only a small variety of practical constraints, which Cinar et al. (2015, p. 34) and Li et al. (2020, p. 73) also experience. Our research contributes by considering a scheduling problem with a large and unique set of practical constraints (e.g., release dates, sequence-dependent changeovers, sequence-dependent job restrictions, transportation times between stages, a limited number of cleanings at the same time, a limited number of shared resources, machine maintenance, and production stops). Moreover, our model allows multiple production routes per job, whereas, to the best of our knowledge, other studies only consider one production route.

Second, Section 3.4 identifies that the majority of the literature concentrates on single objectives. However, single-objectives are insufficient for practical applications as Minella and Ruiz (2008), Lei (2009), and Yenisey and Yagmahan (2014) address. There is a gap in the literature regarding suitable objectives for HFS scheduling problems with practical characteristics (Li, Gao, & Peng, 2020, p. 73). Our research contributes to filling this gap as it considers five objectives and demonstrates the effect of these objectives on several KPIs that express the solution quality (see Section 5.3.1).

Third, to the best of our knowledge, the work of Costa et al. (2020) and Tao et al. (2020) are the only studies that consider additional resources within a production stage in an HFS environment. However, these studies do not consider resources that can be shared over multiple different stages (e.g., IBC-filling, mixing, and packaging) simultaneously. Our research considers a limited number of operators that are required for cleaning machines and we consider IBCs that are shared over all stages. We propose a corrective backtracking algorithm that ensures a limited number of changeovers at any time. This algorithm is applicable when considering, for example, specific limited tooling or operators that are required for operating machines or changeovers.

Fourth, there appears to be a gap in the literature regarding sequencing restrictions between jobs (e.g., the claim constraints) in scheduling problems, which Afzalirad & Rezaeian (2017) also experience. We extend the NEH construction heuristic such that it always obtains feasible solutions regarding the claim constraints.

Finally, this study contributes by demonstrating the effect of several construction heuristics, improvements heuristics, and neighborhood structures on a case study problem with real company data. The evaluation of the results shows that the NEH-construction heuristic with the simulated annealing improvement heuristic and a random neighborhood structure provides good results for our HFS problem with many practical constraints.

References

Afzalirad, M., & Rezaeian, J. (2017). A realistic variant of bi-objective unrelated parallel machine scheduling problem: NSGA-II and MOACO approaches. *Applied Soft Computing, 50*, 109-123.

Al-harkan, I., & Qamhan, A. (2019). Optimize Unrelated Parallel Machines Scheduling Problems With Multiple Limited Additional Resources, Sequence-Dependent Setup Times and Release Date Constraints. *IEEE Access, 7*, 171533-171547.

Alisantoso, D., Khoo, L., & Jiang, P. (2003). An immune algorithm approach to the scheduling of a flexible PCB flow shop. *The International Journal of Advanced Manufacturing Technology, 22*(11-12), 819-827.

Aliyu, A., & Zirra, P. (2013). A Comparative Analysis of Sorting Algorithms on Integer and Character Arrays. *The International Journal Of Engineering And Science*, 25-30.

Blazewicz, J., Ecker, K., Pesch, E., Schmidt, G., & Weglarz, J. (2007). Scheduling with Continuous Resources. In *Handbook on scheduling* (pp. 450-466). Berlin: Springer-Verlag Berlin Heidelberg.

Botta-Genoulaz, V. (2000). Hybrid flow shop scheduling with precedence constraints and time lags to minimize maximum lateness. *International Journal of Production Economics, 64*(1-3), 101-111.

Brucker, P., & Krämer, A. (1996). Polynomial algorithms for resource-constrained and multiprocessor task scheduling problems. *European Journal of Operational Research, 90*(2), 214-226.

Chen, L., Bostel, N., Dejax, P., Cai, J., & Xi, L. (2007). A tabu search algorithm for the integrated scheduling problem of container handling systems in a maritime terminal. *European Journal of Operational Research, 181*(1), 40-58.

Cinar, D., Topcu, Y., & Oliveira, J. (2015). A Taxonomy for the Flexible Job Shop Scheduling Problem. *Optimization, Control, and Applications in the Information Age*, 17-37.

Costa, A., Fernandez-Viagas, V., & Framiñan, J. (2020). Solving the hybrid flow shop scheduling problem with limited human resource constraint. *Computers & Industrial Engineering, 146*, 106545.

Dauzère-Pérès, S., Shen, L., & Neufeld, J. (2018). Solving the flexible job shop scheduling problem with sequence-dependent setup times. *European Journal of Operational Research, 265*, 503-516.

Driessel, R., & Mönch, L. (2011). Variable neighborhood search approaches for scheduling jobs on parallel machines with sequence-dependent setup times, precedence constraints, and ready times. *Computers & Industrial Engineering, 61*, 336-345.

Ebrahimi, M., Fatemi Ghomi, S., & Karimi, B. (2014). Hybrid flow shop scheduling with sequence dependent family setup time and uncertain due dates. *Applied Mathematical Modelling, 38*(9-10), 2490-2504.

Edis, E., Oguz, C., & Ozkarahan, I. (2013). Parallel machine scheduling with additional resources Notation, classification, models and solution methods. *European Journal of Operational Research, 230*(3), 449-463.

Eksioglu, B., Vural, A., & Reisman, A. (2009). The vehicle routing problem: A taxonomic review. *Computers & Industrial Engineering, 57*(4), 1472-1483.

Engin, O., & Doyen, A. (2004). A new approach to solve hybrid flow shop scheduling problems by artificial immune system. *Future Generation Comput Systems, 20*(6), 1083-1095.

Euroma. (2019). *About*. Retrieved from Euroma: https://www.euroma.com/en/about/over-euroma-1

Euroma. (2019). *History*. Retrieved from Euroma: https://www.euroma.com/en/about/Over-Euroma

Euroma. (2019). *Portfolio*. Retrieved from Euroma: https://www.euroma.com/en/about/Nieuws/2019-euroma-move

Gao, J., Gen, M., & Sun, L. (2006). Scheduling jobs and maintenances in flexible job shop with a hybrid genetic algorithm. *Journal of Intelligent Manufacturing, 17*, 493-507.

Gattoufi, S., Oral, M., & Reisman, A. (2004). A taxonomy for data envelopment analysis. *Socio-Economic Planning Sciences, 38*(2-3), 141-158.

Gen, M., & Cheng, R. (1999). *Genetic algorithms and engineering design.* John Wiley & Sons, Inc.

Glover, F., & Laguna, M. (1998). Tabu Search. In D. P. Ding-Zhu, *Handbook of Combinatorial Optimization* (pp. 2093-2229). Boston: Springer US.

Graham, R., Lawler, E., Lenstra, J., & Rinnooy Kan, A. (1979). Optimization and Approximation in Deterministic Sequencing and Scheduling: a Survey. *Annals of Discrete Mathematics, 5*, 287-326.

Guinet, A., & Solomon, M. (1996). Scheduling hybrid flowshops to minimize maximum tardiness or maximum completion time. *International Journal of Production Research, 34*(6), 1643-1654.

Heerkens, J., & Winden, A. v. (2017). *systematisch management problemen oplossen.* Groningen/Houten: Noordhof Uitgevers.

Jin, Z., Yang, Z., & Ito, T. (2006). Metaheuristic algorithms for the multistage hybrid flowshop scheduling problem. *International Journal of Production Economics, 100*(2), 322-334.

Jun, S., & Park, J. (2015). A hybrid genetic algorithm for the hybrid flow shop scheduling problem with nighttime work and simultaneous work constraints: A case study from the transformer industry. *Expert Systems with Applications, 42*(15-16), 6196-6204.

Karimi, S., Ardalan, Z., Naderi, B., & Mohammadi, M. (2017). Scheduling flexible job-shops with transportation times: Mathematical models and a hybrid imperialist competitive algorithm. *Applied Mathematical Modelling, 41*, 667-682.

Kirkpatrick, S. (1983). Optimization by Simulated Annealing. *Science*, 671-680.

Kupfahl, J., & Bierwirth, C. (2016). A study on local search neighborhoods for the job shop scheduling problem with total weighted tardiness objective. *Computers & Operations Research, 66*, 44-57.

Lei, D. (2009). Multi-objective production scheduling: a survey. *The International Journal of Advanced Manufacturing Technology, 43*(9), 926-938.

Li, J., Duan, P., Sang, H., Wang, S., Liu, Z., & Duan, P. (2018). An effiecient algorithm for resource-constrained steelmaking scheduling problems. *IEEE Access*, 33883-33894.

Li, X., Gao, L., & Peng, K. (2020). Review for flexible job shop scheduling. In *Effective Methods for Integrated Process Planning and Scheduling* (pp. 17-45). Berlin, Heidelberg: Springer.

Lin, S., & Ying, K. (2013). Minimizing makespan and total flowtime in permutation flowshops by a bi-objective multi-start simulated-annealing algorithm. *Computers & Operations Research, 40*(6), 1625-1647.

Linn, R., & Zhang, W. (1999). Hybrid flow shop scheduling: a survey. *Computers & Industrial Engineering, 37*(1-2), 57-61.

Linn, R., & Zhang, W. (1999). Hybrid flow shop scheduling: A survey. *Computers & Industrial Engineering, 37*(1-2), 57-61.

Liu, W., Jin, Y., & Price, M. (2017). A new improved NEH heuristic for permutation flowshop scheduling problems. *International Journal of Production Economics*, 21-30.

Low, C. (2005). Simulated annealing heuristic for flow shop scheduling problems with unrelated parallel machines. *Computers & Operations Research, 32*(8), 2013-2025.

Minella, G., & Ruiz, R. (2008). A Review and Evaluation of Multiobjective Algorithms for the Flowshop Scheduling Problem. *Journal on Computing, 20*(3), 451-471.

Naderi, B., Zandieh, M., & Roshanaei, V. (2009). Scheduling hybrid flowshops with sequence dependent setup times to minimize makespan and maximum tardiness. *The International Journal of Advanced Manufacturing Technology, 41*, 1186-1198.

Naderi, B., Zandieh, M., Khaleghei, A., & Roshanaei, V. (2009). An improved simulated annealing for hybrid flowshops with sequence-dependent setup and transportation times to minimize total completion time and total tardiness. *Expert Systems with Applications, 36*, 9625-9633.

Nawaz, M., Enscore, E., & Ham, I. (1983). A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. *Omega, 11*(1), 91-95.

Oğuz, C., Zinder, Y., Do, V., Janiak, A., & Lichtenstein, M. (2004). Hybrid flow-shop scheduling problems with multiprocessor task systems. *European Journal of Operational Research, 152*(1), 115-131.

Pinedo, M. (2016). *Scheduling.* 2016: Springer Cham Heidelberg New York.

Pinto, J., & Grossmann, I. (1998). Assignment and sequencing models for scheduling of process systems. *Annals of Operations Research*(81), 433-466.

Quadt, D., & Kuhn, H. (2007). A taxonomy of flexible flow line scheduling procedures. *European Journal of Operations Research, 178*(3), 686-698.

Rajagopalan, D., & Karimi, I. (1989). Completion time in serial mixed-storage multiproduct processes with transfer and set-up times. *Computers Chemical Engineering, 13*(1/2), 175-186.

Reisman, A., Kumar, A., & Motwani, J. (1997). Flowshop scheduling/sequencing research: a statistical review of the literature, 1952-1994. *IEEE Transactions on Engineering Management, 44*(3), 316-329.

Riane, F., Artiba, A., & Elmaghraby, S. (2002). Sequencing a hybrid two-stage flowshop with dedicated machines. *International Journal of Production Research, 40*(17), 4353-4380.

Ribas, I., Leisten, R., & Framiñan, J. (2010). Review and classification of hybrid flow shop scheduling problems from a production system and a solutions procedure perspective. *Computers & Operations Research, 37*(8), 1439-1454.

Romero, C. (1991). *Handbook of critical issues in goal programming.* Córdoba: Elsevier.

Ruiz, R., & Maroto, C. (2006). A genetic algorithm for hybrid flowshops with sequence dependent setup times and machine eligibility. *European Journal of Operational Research, 169*(3), 781-800.

Ruiz, R., & Vázquez-Rodríguez, J. (2010). The Hybrid Flow Shop Scheduling Problem. *European Journal of Operational Research, 205*(1), 1-18.

Ruiz, R., Sivrikaya Şerifoğlu, F., & Urlings, T. (2008). Modeling realistic hybrid flexible flowshop scheduling problems. *Computers & Operations Research, 35*(4), 1151-1175.

Sun, L., Cheng, X., & Liang, Y. (2010). Solving Job Shop Scheduling Problem Using Genetic Algorithm with Penalty Function. *International Journal of Intelligent Information Processing, 1*(2), 65-77.

Tahmasbi, H., & Moghaddam, R. (2011). Solving a bi-objective flowshop scheduling problem by a Multi-objective Immune System and comparing with SPEA2+ and SPGA. *Advances in Engineering Software, 42*(10), 772-779.

Tang, L., Liu, W., & Liu, J. (2005). A neural network model and algorithm for the hybrid flow shop scheduling problem in a dynamic environment. *Journal of Intelligent Manufacturing, 16*(3), 361-370.

Tang, L., Zhang, Y., Wang, J., Liao, X., & Yi, Z. (2005). Heuristic combined artificial neural networks to schedule hybrid flow shop with sequence dependent setup times. In *Advances in Neural Networks - ISNN 2005* (pp. 788-793). Berlin: Springer Berlin Heidelberg.

Tao, X., Li, J., Huang, T., & Duan, P. (2020). Discrete imperialist competitive algorithm for the resource-constrained hybrid flowshop problem with energy consumption. *Complex & Intelligent Systems*, 311-326.

T'Kindt, V., & Billaut, J. (2006). *Multicriteria Scheduling.* Berlin Heidelberg: Springer.

Tseng, C., Liao, C., & Liao, T. (2008). A note on two-stage hybrid flowshop scheduling with missing operations. *Computers & Industrial Engineering, 54*(3), 695-704.

Voß, S., & Witt, A. (2007). Hybrid flow shop scheduling as a multi-mode multi-project scheduling problem with batching requirements: A real-world application. *International Journal of Production Economics, 105*(2), 445-458.

Wang, H. (2005). Flexible flow shop scheduling: Optimum, heuristics and artificial intelligence solutions. *Expert Systems, 22*(2), 78-85.

Wang, H., Jacob, V., & Rolland, E. (2003). Design of efficient hybrid neural networks for flexible flow shop scheduling. *Expert Systems, 20*(4), 208-232.

Wang, X., & Tang, L. (2009). A tabu search heuristic for the hybrid flowshop scheduling with finite intermediate buffers. *Computers & Operations Research, 36*(3), 907-918.

Yaurima, V., Burtseva, L., & Tchernykh, A. (2009). Hybrid flowshop with unrelated machines, sequence-dependent setup time, availability constraints and limited buffers. *Computers & Industrial Engineering, 56*(4), 1452-1463.

Yenisey, M., & Yagmahan, B. (2014). Multi-objective permutation flow shop scheduling problem: Literature review, classification and current trends. *Omega, 45*, 119-135.

Ying, K., & Lin, S. (2009). Scheduling multistage hybrid flowshops with multiprocessor tasks by an effective heuristic. *International Journal of Production Research, 47*(13), 3525-3538.

Zhang, G., Zhang, L., Song, X., Wang, Y., & Zhou, C. (2019). A variable neighborhood search based genetic algorithm for flexible job shop scheduling problem. *Cluster Computing, 22*, 11561-11572.

Appendices

Appendix 1 General production process overview

Figure 37 provides a general overview of the production processes of the facility in Zwolle. Note that the process stages marked in blue are not in the scope of this research. Sections 2.1.1, 2.1.2, and 2.1.3 explain respectively the replenishing, mixing, and packaging processes. The colors that correspond to these processes are used in all other process flows accordingly. The grey boxes illustrate the departments and their process stages.

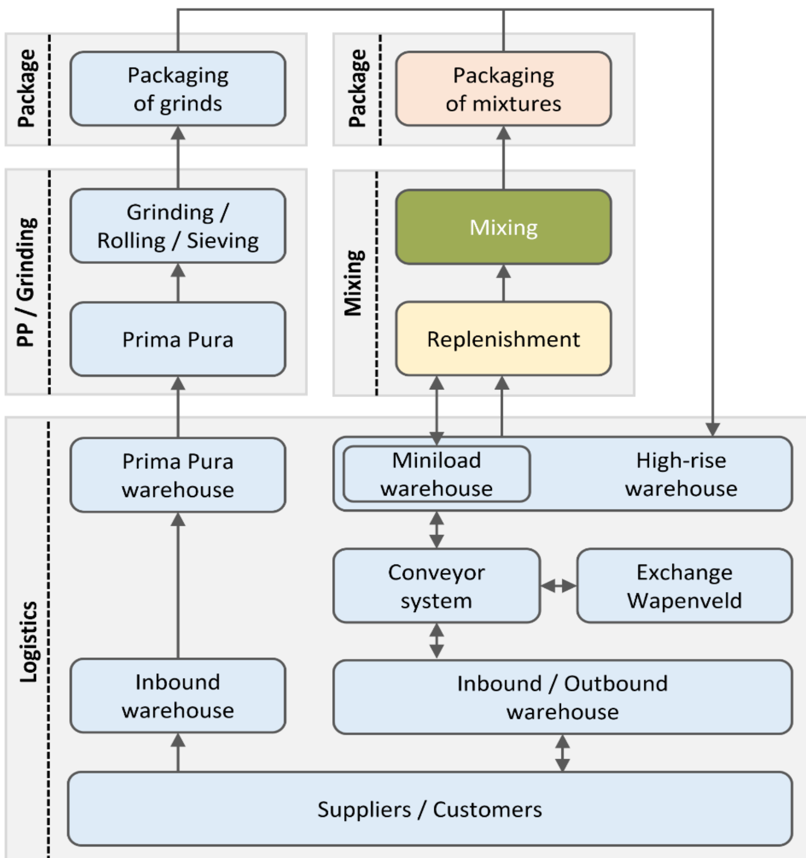


Figure 37 | General production process flow

The logistics department handles all the inbound flow from the suppliers and other facilities of Euroma as well as the outbound flow to the customers. There are two warehouses for the inbound and outbound flow of which one warehouse is dedicated to untreated raw materials. The Quality Officers inspect the untreated raw materials before transportation to the Prima Pura warehouse. The warehouse for the inbound and outbound flow connects via a conveyor system to the high-rise warehouse. This high-rise warehouse is fully automated and holds 22,000 pallet locations. Within this high-rise warehouse is the miniload warehouse with a capacity of 2,000 tote locations.

The Prima Pura and grinding departments receive the untreated raw materials from the Prima Pura warehouse. The Prima Pura installation steams these raw materials to increase the safety, quality, and shelf-life. After the Prima Pura process, the raw materials are often ground or rolled, and sieved. Dedicated packaging lines package the treated raw materials onto pallets and operators store these pallets into the high-rise warehouse.

The mixing department receives raw materials from the high-rise warehouse and the miniload warehouse. First, the replenishment process collects all raw materials that are in the recipe of a mixture. After that, the mixing process starts. After the mixing process, dedicated packaging lines package the mixtures onto pallets and operators store these pallets into the high-rise warehouse.

Appendix 2 Packaging process

Figure 38 provides the discharging and packaging process flow. The process stages within the packaging department are marked in light-orange and the transportation mediums are color-marked according to the legend.

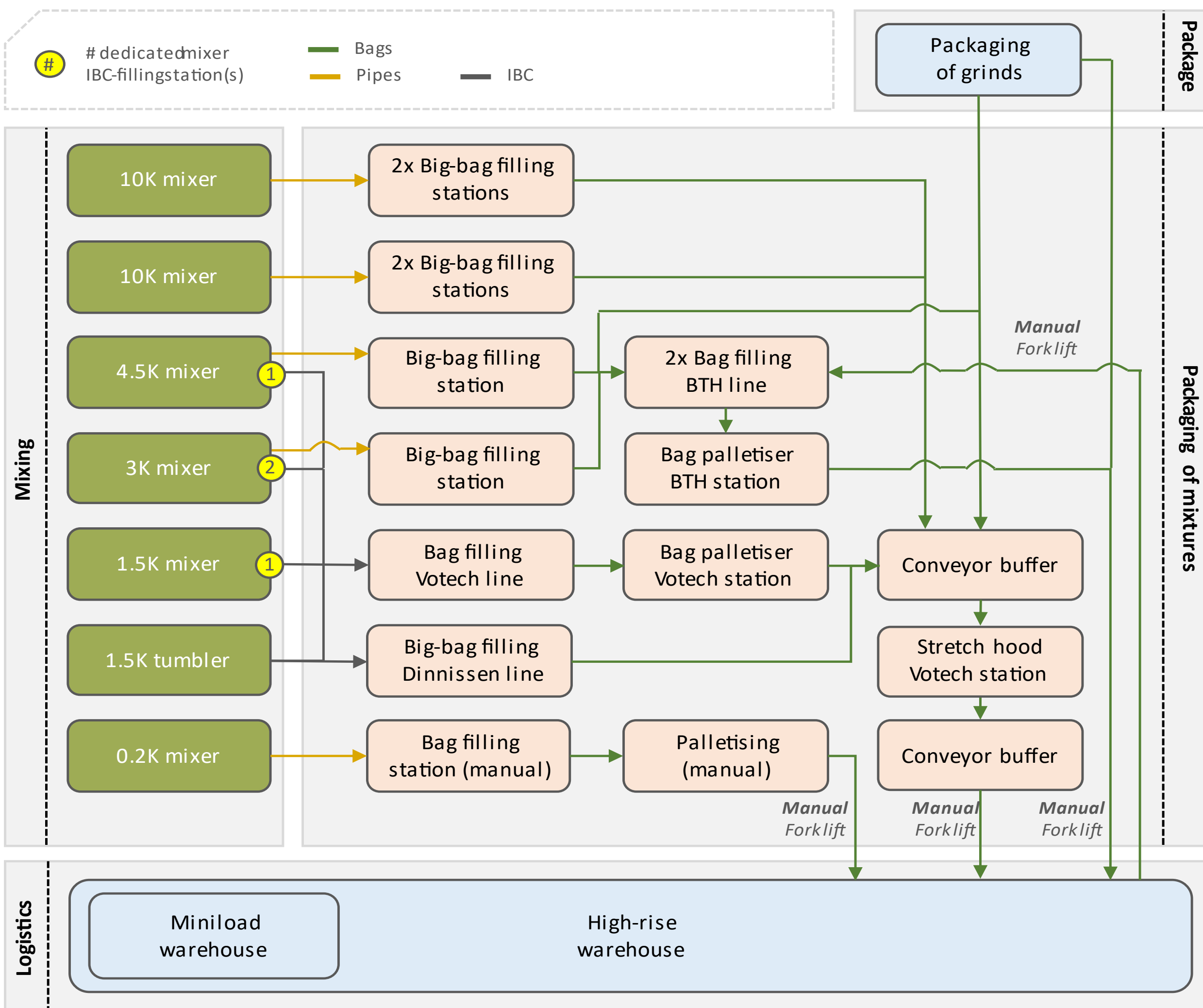


Figure 38 | Packaging process flow

From Figure 38 we note that the two 10K mixers each have two discharging stations that fill big bags. The 1.5K, 3K, and 4.5K mixers each have one dedicated discharging station to fill IBC and one dedicated discharging station to fill big bags. However, these discharging stations are for internal use only as they do not contain a sieve. The 0.2K mixer discharges manually with the help of an operator. Further, all discharging stations have a weighing system that weighs the requested amount. After discharging, AGVs or operators transport the mixture to the high-rise warehouse or to a packaging line. Figure 39 provides a Sankey diagram of the flows from the mixers to the final packaging units. The numbers correspond with the number of production routes, e.g., both 10K mixers have 2 routes and there are 11 possible routes to have the final packaging unit in bags. Figure 38

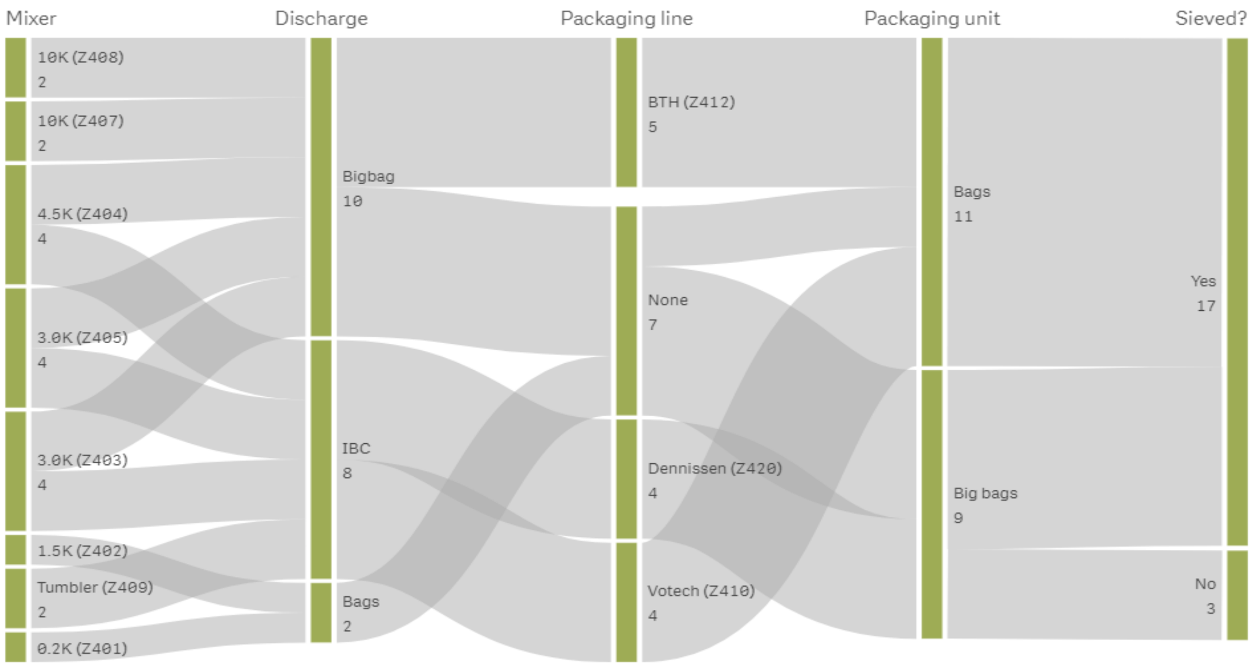


Figure 39 | Sankey diagram of the flow from the mixers to the final packaging units

All packaged mixtures are palletized, e.g., big bags are palletized at the big-bag filling stations and the bags are palletized at dedicated palletizing stations. Figure 40 depicts an example of a palletizer and Figure 41 shows the stretch hood machine that wraps and labels pallets. All pallets, except the pallets from the BTH line are wrapped and labeled in a stretch hood machine. After wrapping and labeling, a manual forklift puts the pallets onto the conveyor belt of the high-rise warehouse.

Figure 40 | Palletizer

Figure 41 | Stretch hood

Appendix 3 Production routes

A route is a sequence of production machines to produce a product. Thus, a route determines which machines and resources (e.g., IBCs) to use. Products can have multiple production route options. Further, all products have a default production route, i.e., the routes that the ERP system suggests according to the input of a process engineer.

There are a few criteria that determine which production routes are suitable. These criteria are: (i) the product weight and density, (ii) if the recipe requires cutting blades in a mixer, (iii) if the recipe requires liquid, and (iv) the final packaging unit.

First, each mixer can only mix a product that is within a certain range of weight and volume, e.g., a mixer with the capacity of 10,000L cannot mix a product with a volume of 2000L, as the mixing blades are not effective when the mixer is filled for 20%. Second, some mixers have cutting blades to cut for example vegetables for soups. Recipes that require these cutting blades can only be allocated to mixers that have these cutting blades. Third, for some mixers, it is not possible to add liquids. Thus, recipes that require liquids can only be allocated to mixers that have the option to add liquids. Fourth, the final packaging unit (e.g., big-bag or bag) determines which packaging line is suitable.

The route determines the machines and the resources. For example, consider a mixture with a volume of 4500L that is suitable for mixing on the 4.5K and the 10K mixers. The 10K mixer can replenish raw materials directly from the outdoor silos and IBCs, whereas the 4.5K mixer can only replenish raw materials from IBCs. In this case, the 10K mixer may require only one IBC for replenishment and the 4.5K mixer may require 4 IBCs. As a result, the number of IBCs to replenish affects the processing time at the IBC-filling stations. Thus, the route affects the usage of resources.

Appendix 4 A detailed description of the scheduling problem taxonomic framework

This section describes all attributes from the taxonomic framework in Table 30 that we combine in Section 3.2 based on several common scheduling problem attributes from the literature. The remainder of this section elaborates upon these attributes per field.

The α-field of the taxonomy defines the machine environment and the maintenance policies. Section 3.1 describes the different machine environments. The maintenance policy of machines can be variable, fixed, or non-existent. A maintenance policy is variable when the starting times of maintenance activities are flexible. It is fixed if the starting times of maintenance activities are predefined (Cinar, Topcu, & Oliveira, 2015, p. 29).

Next, the β-field defines the job characteristics, sequencing relations, transportation- and inventory policies, and other characteristics. Processing times of jobs can be fixed or dependent on the operation, stage, or machine (Cinar, Topcu, & Oliveira, 2015, p. 27). Moreover, jobs may have release times. In that case, these jobs cannot start before their release time (Cinar, Topcu, & Oliveira, 2015, p. 29). Alternatively, jobs may also have due dates. In that case, jobs should not finish later than their due date (Ribas, Leisten, & Framiñan, 2010).

Scheduling problems can have several characteristics that depend on the job sequence. For instance, changeovers between two consecutive jobs on the same machine can depend on the job sequence, the machine, the time, or the frequency of utilization (Pinto & Grossmann, 1998, p. 437). Changeovers are sequence-dependent when different pairs of consecutive jobs result in different changeovers (Cinar, Topcu, & Oliveira, 2015, p. 29). These changeovers may also depend on the machine (Ribas, Leisten, & Framiñan, 2010, p. 1442). For instance, two different machines that produce the same job sequence may require different changeovers. Moreover, changeovers can depend on the frequency of utilization arise, for instance, when a machine requires a changeover after processing every predetermined quantity of units. Thus, more changeovers are required in case a machine produces more. Time dependent cleaning arises when a machine requires a changeover after a certain time interval (Pinto & Grossmann, 1998, p. 438). Changeovers can also be independent and fixed, for instance, a changeover with a fixed time can be required between every pair of consecutive jobs (Ribas, Leisten, & Framiñan, 2010).

Furthermore, some scheduling problems consider sequencing constraints between the operations of a job or between jobs. The most common sequencing constraint ensures that the operations of a job produce in a predefined order. Alternatively, when there are no sequencing constraints, the operations of a job can produce in parallel in an arbitrary order (Pinto & Grossmann, 1998, p. 435).

Some scheduling problems consider transportation times between stages or machines. Transportation times are also referred to as transfer-, or removal times. These times can be fixed or variable. For the latter. the transportation time depends on the start- and endpoint (Ribas, Leisten, & Framiñan, 2010, p. 1442). Moreover, transportation times can be asymmetric, i.e., the transportation time from A to B differs from the time from B to A.

The following three different types of intermediate storage policies are common: no-intermediate storage (NIS), finite-intermediate storage (FIS), and unlimited-intermediate storage (UIS) (Rajagopalan & Karimi, 1989).

Table 30 | Scheduling problem classification framework

Machine environment (α)		
Machine characteristics	Machine environment	Single machine
		Parallel machines
		Flow shop
		Flexible flow shop
		Hybrid flow shop
		Job shop
		Flexible job shop
		Open shop
	Maintenance policy	Variable
		Fixed
		None
Job characteristics & constraints (β)		
Job characteristics	Processing time	Operation-dependent
		Stage-dependent
		Machine-dependent
		Fixed
	Release dates	For jobs
		None
	Due dates	For jobs
		None
Sequence	Changeovers	Sequence-dependent
		Machine-dependent
		Time-dependent
		Frequency-dependent
		Fixed
		None
	Sequencing constraints	Between jobs
		Between operations
		None
Transport	Transportation times	Variable
		Fixed
		None
	Inventory policy	Unlimited
		Finite
Other characteristics	Demand pattern	Variable
		Fixed (cyclic)
	Time representation	Continuous
		Discrete (fixed slots)
	Resource constraints	Continuous
		Discrete
		None (only machines)
	Lot splitting	Over machines
		None
Objective function (γ)		
Objective	Objective	Makespan
		Flowtime
		Tardiness
		Earliness
		Costs
		Other

For the NIS policy, there is no storage capacity between stages. Therefore, jobs need to wait on the machine until the next machine is available or they need to transfer to the next stage immediately upon completion. The FIS policy has a finite storage capacity, and the UIS policy has unlimited storage capacity. In practice, FIS is the most common inventory policy (Pinto & Grossmann, 1998, p. 437). Note that the FIS policy can include both the NIS and UIS policies by simply setting the storage capacity to zero or infinity, respectively (Rajagopalan & Karimi, 1989).

Demand patterns in scheduling problems can be variable or fixed. Variable demand occurs when products have little similarity between demand patterns in different scheduling periods. On the contrary, demand can be considered fixed in case products have constant demand rates. Fixed demand is often referred to as cyclic demand, as the same production sequence can be scheduled repeatedly. Variable demand scheduling often has a short-term horizon, whereas fixed demand scheduling is often considered with a more long-term horizon (Pinto & Grossmann, 1998, p. 437).

The time domain representation in scheduling problems can either be discrete or continuous. A discrete time representation consists of time slots. Time slots have equal and fixed intervals for unit allocation. The duration of time slots need to be sufficiently small to have a suitable approximation of the problem. Often, the time slot duration is set to be the greatest common factor of the processing times of the jobs. In the case of a discrete formulation, some time slots may remain empty as not every time slot has a unit allocation. When the time representation is continuous, unit allocations are associated with time events instead of time slots (Pinto & Grossmann, 1998, p. 436).

In scheduling problems, different operations can require the same resources (e.g., utilities or manpower). In general, these resources have a finite capacity. Therefore, it is necessary to consider feasible combinations of operations that do not exceed the limited resource availability. These resources can be considered discrete when the consumption is at a constant level during the process, or continuous, in which the resource consumption differs during the process (Pinto & Grossmann, 1998, p. 437). The resource capacity needs to be sufficient to ensure feasibility (Brucker & Krämer, 1996).

Some scheduling problems consider lot-sizing. Lot-sizing allows to divide the workload of a job over multiple identical machines. However, most scheduling problems do not allow lot-sizing as they consider that each job can only produce on one machine at a time (Ribas, Leisten, & Framiñan, 2010, p. 1443).

Finally, the γ-field defines the objective function criteria. Scheduling problems can have single- or multiple objective functions. Some common objectives are shown in Table 8. In scheduling, the makespan refers to the difference between the start- and finish times of a schedule. The flowtime of a job is the difference between the completion- and the release times of a job. Tardiness refers to the time a job is completed after its due date (Graham, Lawler, Lenstra, & Rinnooy Kan, 1979). Accordingly, the earliness is the time that jobs complete before their due date. Moreover, objectives often include production costs (e.g., operating costs, transportation costs, or penalty costs for completing jobs after their due dates) (Cinar, Topcu, & Oliveira, 2015, p. 29).

Appendix 5 Calculating the number of IBCs needed

Recall from Section 4.4 that max^{IBC} is the maximum allowed number of IBCs needed (i.e., in production and cleaning) at any time. Besides that, let the number of IBCs for replenishing and discharging the operation $o_{j,r,i}$ on machine $k \in M_{j,r,i}$ be $IBC^{in}_{j,r,i,k}$ and $IBC^{out}_{j,r,i,k}$, respectively. The number of IBCs needed can vary during an operation, e.g., more IBCs can be needed for filling than for discharging a machine (see Section 2.1.3). In the case that $IBC^{in}_{j,r,i,k} > IBC^{out}_{j,r,i,k}$, the number of IBCs that leave the production status (i.e., $IBC^{in}_{j,r,i,k} - IBC^{out}_{j,r,i,k}$), become available consecutively after filling machine k for a duration of $filltime^{IBC}_{k}$. An IBC that leaves the production process changes its status to cleaning. At first, this IBC needs to transport from machine $k \in M$ to the IBC-cleaning stations with a duration of $transp^{IBC}_{k}$. A single IBC-cleaning station can clean one IBC at a time with a duration that is denoted by $cleaning^{IBC}$. Finally, let IBC^{prev} be the number of IBCs at the IBC-cleaning station at the start of the new schedule.

Figure 42 illustrates the IBC-usage of a job. The Gantt chart shows the operations on the machines and the stacked line chart shows the status of the IBCs over time. Regarding the latter, the blue and the red area indicate, respectively, the number of IBCs in production and cleaning.

We enumerate the number of IBCs in production and in cleaning every time that an IBC changes its status. An IBC changes its status once it (i) starts production, (ii) finishes production and starts cleaning, and (iii) finishes cleaning. To illustrate this, consider the example in Figure 42. The first operation needs four IBCs during the whole process ($IBC^{in} = IBC^{out} = 4$). The second operation requires four IBCs for replenishing and three IBCs for discharging ($IBC^{in} = 4,\ IBC^{out} = 3$). Therefore, the first IBC replenishes machine k with a duration of $filltime^{IBC}_{k} = 10$, and after that, this IBC changes its status from production to cleaning. Subsequently, this IBC transports from machine k to the IBC-cleaning station with a duration of $transp^{IBC}_{k} = 10$. When the IBC arrives at the IBC-cleaning station, it enters the queue for cleaning with a duration of $cleaning^{IBC} = 10$. After cleaning, the IBC becomes available again for production. The remaining three IBCs that are still in production fill the machine for the third operation and then transport to the IBC-cleaning station consecutively. Note that there are five IBCs in cleaning at the start of the new production schedule ($IBC^{prev} = 5$). These IBCs form a queue at the cleaning station. Therefore, these IBCs are clean after one hour ($transp^{IBC}_{k} + 5 \cdot cleaning^{IBC} = 10 + 50$).

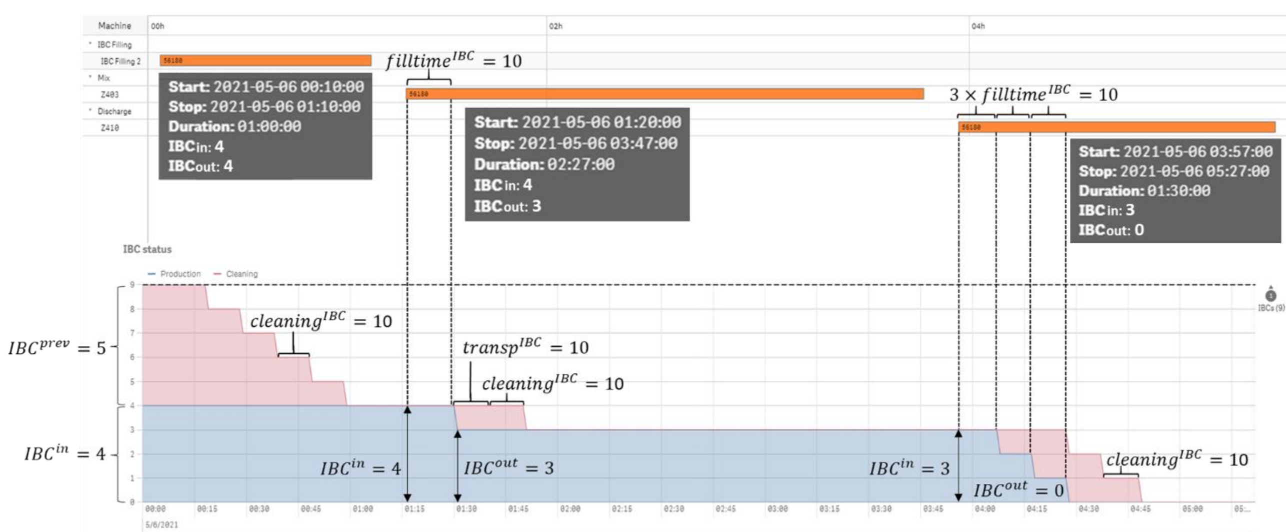


Figure 42 | Example of IBC-usage

Appendix 6 Random construction heuristic pseudo code

Section 4.6.1 describes a random construction heuristic (RCH) that randomly generates a solution for the purpose to benchmark the performance of the improvement heuristics. We extend this heuristic such that it can deal with sequencing constraints (e.g., the claim constraints as described in Section 2.2.2). Figure 43 provides a pseudo-code of the RCH for the purpose to enhance the model's credibility and reproducibility.

```
Random construction heuristic
Initialize JobsToSchedule, CountInfeasible ← 0
While JobsToSchedule.Count > 0 do
    j ← GetRandom(JobsToSchedule)
    If Job[j].NrScheduled = 0 then
        r ← GetRandom(Job[j].Route) else r ← Job[j].OnRoute
    End
    Operation ← Job[j].NrScheduled
    For i = Operation to Job[j].Route[r].Operation.Count do
        k ← GetRandom(Job[j].Route[r].Operation[i].Machine)
        pos ← GetRandom(V^s)
        V^s.Insert(index = pos, value= j)
        V^m.Insert(index = pos, value= k)
        If ClaimConstraintsSatisfied(k, pos) then
            Increment(Job[j].NrScheduled)
            If t = Job[j].Route[r].Operation.Count then
                JobsToSchedule.Delete(j)
            End
        ElseIf CountInfeasible = Limit then
            pos ← V^s.Count
            V^s.Insert(indexRange = [pos, pos + 1], value= dummy)
            V^m.Insert(indexRange = [pos, pos + 1], value= k)
            V^s.Insert(index = pos + 2, value= j)
            V^m.Insert(index = pos + 2, value= k)
            CountInfeasible ← 0
        Else
            Increment(CountInfeasible)
            V^s.Delete(index = pos)
            V^m.Delete(index = pos)
            Break for loop
        End
    End
End
DecodeSchedule(V^s, V^m)
```

Figure 43 | Pseudo-code of the random construction heuristic

Appendix 7 NEH construction heuristic pseudo code

Section 4.6.2 describes the NEH construction heuristic that we extend such that it can deal with sequencing constraints. Figure 44 provides a pseudo-code of this heuristic for the purpose to enhance the model's credibility and reproducibility.

Extended NEH construction heuristic

```
Initialize JobsToSchedule.Sort.Descending(TAPT_j), item ← 0
While JobsToSchedule.Count > 0 do
    j ← JobsToSchedule[item]
    If Job[j].NrScheduled = 0 then
        r ← GetRandom (Job[j].Route) else r ← Job[j].OnRoute
    EndIf
    Initialize BestObjective ← ∞, FoundFeasible ← False, i ← Job[j].NrScheduled
    For k in Job[j].Route[r].Operation[i].Machine do
        For pos in V⃗^s do
            V⃗^s.Insert(index = pos, value= j)
            V⃗^m.Insert(index = pos, value= k)
            If ClaimConstraintsSatisfied(k, pos) then
                CurrObjective ← CalculateObjective(V⃗^s, V⃗^m)
                FoundFeasible ← True
                If CurrObjective < BestObjective then
                    BestObjective ← CurrObjective,  Best_k ← k, Best_pos ← pos
                EndIf
            V⃗^s.Delete(index = pos)
            V⃗^m.Delete(index = pos)
        End
    End
    If FoundFeasible then
        V⃗^s.Insert(index = Best_pos, value= j)
        V⃗^m.Insert(index = Best_pos, value= Best_k)
        Increment(Job[j].NrScheduled), item ← 0
        If Job[j].NrScheduled = Job[j].Route[r].Operation.Count then
            JobsToSchedule.Delete(item)
        EndIf
    ElseIf item = JobsToSchedule.Count then
        pos ← V⃗^s.Count
        V⃗^s.Insert(indexRange = [pos, pos + 1], value= dummy)
        V⃗^m.Insert(indexRange = [pos, pos + 1], value= k)
        V⃗^s.Insert(index = pos + 2, value= j)
        V⃗^m.Insert(index = pos + 2, value= k)
    Else
        Increment(item)
    EndIf
End
DecodeSchedule(V⃗^s, V⃗^m)
```

Figure 44 | Pseudo code extended NEH construction heuristic

Appendix 8 Neighborhood structure parameters tuning

This section describes the tuning process of the neighborhood structure that makes the tradeoff between diversifying and intensifying the neighborhood search. Recall that this neighborhood structure first intensively searches the current search space by using the small neighborhood operators. After having searched the current space and the limit on the number of attempts for finding a better solution ($count_{stop}$) is reached, we use the operator N4 (change the production route) to identify a new search space. Therefore, it is necessary to determine the parameter $count_{stop}$. To achieve this, we first consider the following values for $count_{stop}$ {25; 50; 75; 100; 125; 150; 175; 200}. For each $count_{stop}$ scenario, we conduct 3 replications (i.e., solving every scenario 3 times with the same parameter settings to provide statistically significant results). In each replication, we use the NEH construction heuristic and the SA improvement heuristic of which the parameter values are described in Appendix 9. Table 31 provides the experiments including the CPU time and the objective value per scenario, of which the standard deviation is reported between brackets.

Table 31 | Neighborhood structure parameter tuning

Scenario	$count_{stop}$	Objective	CPU time (s)
1	25	7713.3 (95.7)	107.3 (1.2)
2	50	7964.7 (131.9)	107.7 (0.5)
3	75	7903.0 (65.5)	108.5 (0.8)
4	100	7906.0 (87.3)	114.0 (3.9)
5	125	7982.7 (199.9)	112.5 (6.1)
6	150	7997.7 (53.1)	110.5 (0.6)
7	175	8145.3 (62.9)	108.5 (0.6)
8	200	8272.7 (150.8)	109.6 (6.8)

Following from the results from Table 31, we observe that scenario 1, which has the lowest $count_{stop}$ value provides the best objective values. To evaluate whether $count_{stop} = 25$ provides the best results concerning the objective value, we perform an extra tuning experiment with the following values for $count_{stop}$ {5; 10; 15; 20; 25; 30; 35}. The results of these experiments are in Table 32.

Table 32 | Neighborhood structure detailed parameter tuning

Scenario	$count_{stop}$	Objective	CPU time (s)
1	5	7615.0 (87.3)	111.7 (1.8)
2	10	7690.7 (23.5)	115.0 (0.8)
3	15	7649.3 (40.0)	111.4 (3.3)
4	20	7710.7 (130.8)	113.4 (4.8)
5	25	7725.7 (93.1)	114.2 (4.0)
6	30	7954.7 (126.3)	116.0 (1.5)
7	35	7783.7 (162.5)	116.8 (0.7)

From the results in Table 32, we choose to select the parameter values of scenario 1, as these provide the best objective values.

Appendix 9 Simulated annealing parameters tuning

This section describes the tuning process of the simulated annealing (SA) parameters. At first, we choose to determine the start temperature of the heuristic by solving instance 3 (see Section 5.1) with SA with a starting temperature of 250. We set the Markov chain length at 500 and the cooling factor at 0.99. Every time the Markov chain is at its end, we store the acceptance ratio (i.e., the number of accepted neighbors divided by the number of generated neighbors) of that Markov chain. The results are in Figure 45.

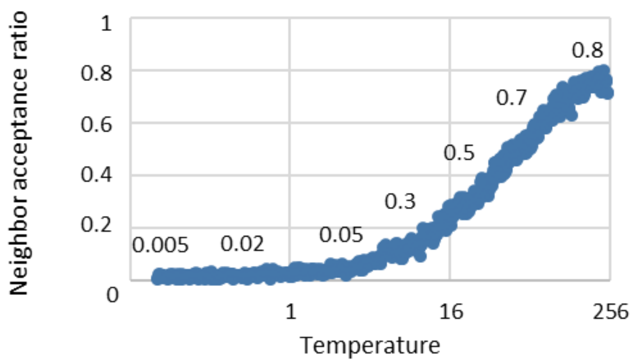


Figure 45 | Neighbor acceptance ratio per temperature

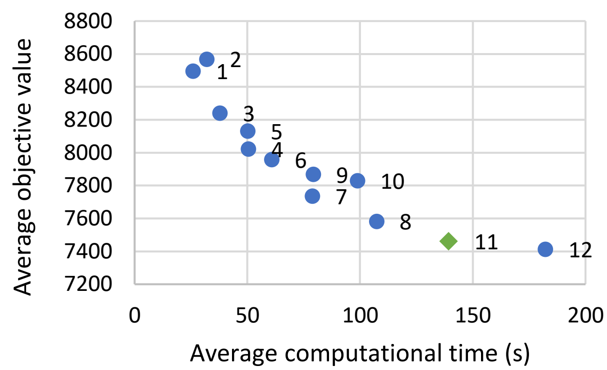


Figure 46 | Computational time compared to the objective value per scenario

We choose to set the starting temperature at 100, which results in an acceptance ratio of about 0.75. We choose not to start with a larger acceptance ratio since the NEH construction heuristic generates good initial solutions. Therefore, we argue that a large acceptance ratio might worsen the initial solution in the beginning, resulting in computational waste.

Next, we consider the following Markov chain length values {250; 500; 750} and the following values for the decrease factor {0.985; 0.9875; 0.99; 0.9925}. For each combination of the Markov chain length and decrease factor, we conduct 3 replications (i.e., solving every scenario 3 times with the same parameter settings to provide statistically significant results). Table 33 provides the experiments including the CPU time and the objective value per scenario, of which the standard deviation is reported between brackets. Moreover, Figure 46 depicts the results. From these results, we choose to select the parameter values of scenario 11, as these give a good compromise between the CPU time and the solution quality.

Table 33 | Simulated annealing parameter tuning

Scenario	Start temperature	Markov chain length	Decrease factor	Objective	CPU time (s)
1	100	250	0.985	8497.0 (209.2)	30.4 (0.7)
2	100	250	0.9875	8568.7 (178.4)	36.6 (2.1)
3	100	250	0.99	8241.7 (164.1)	42.3 (0.1)
4	100	250	0.9925	8023.0 (82.6)	54.8 (0.3)
5	100	500	0.985	8131.0 (100.5)	54.5 (0.1)
6	100	500	0.9875	7959.3 (111.8)	65.2 (0.3)
7	100	500	0.99	7735.7 (105.9)	83.4 (1.8)
8	100	500	0.9925	7583.0 (120.5)	112.0 (0.8)
9	100	750	0.985	7868.3 (192.9)	83.8 (1.1)
10	100	750	0.9875	7829.0 (155.0)	103.7 (7.4)
11	100	750	0.99	7460.7 (93.9)	144.4 (1.2)
12	100	750	0.9925	7413.7 (122.4)	187.5 (4.6)

Appendix 10 Tabu lists length tuning

This section describes the tuning process of the tabu list length. As we consider a tabu list for each neighborhood operator, we should set the tabu list length for the (i) swap operator, (ii) move operators, and (iii) the change production route operator.

There are many different swap- and move neighbors. However, the number of jobs that can change the production route is limited. Therefore, we consider the same tabu list length for the swap and move operators and a separate tabu list length for the change production route operator. The tabu list lengths that we consider are {25; 50; 75}. For each tabu list length configuration scenario, we conduct 3 replications (i.e., solving every scenario 3 times with the same parameter settings to provide statistically significant results). Table 34 provides the full factorial experiment including the CPU time and the objective value per scenario, of which the standard deviation is reported between brackets.

Table 34 | Tabu list length parameter tuning

Scenario	Move & Swap tabu list length	Change route tabu list length	Objective	CPU time (s)
1	25	25	7738.7 (15.5)	91.7 (1.1)
2	25	50	7731.7 (17.5)	91.5 (1.1)
3	25	75	7819.0 (96.0)	91.2 (1.5)
4	50	25	7726.0 (40.4)	91.1 (1.2)
5	50	50	7609.0 (145.2)	91.1 (0.4)
6	50	75	7815.0 (165.0)	91.3 (0.7)
7	75	25	7786.7 (237.2)	90.6 (0.7)
8	75	50	7532.7 (174.0)	92.1 (0.6)
9	75	75	7831.3 (54.5)	91.2 (0.9)

From the results in Table 34, we choose to select the parameter values of scenario 8, as these provide the best objective values. Note that the length of the change production route tabu list is lower than the length of the swap and move tabu lists. This might be due to the limited number of neighbors for the change route operator compared to the other operators. Also, note that the CPU time does not vary between the scenarios. A possible explanation is that when the tabu list lengths are short (e.g., scenario 1), extra time is spent on evaluating neighbors that have been visited twice and less time is spent on checking the tabu lists for tabu neighborhood operators. On the other hand, scenarios with long tabu lists (e.g., scenario 9) might spend less time on evaluating neighbor solutions that have been visited twice and more time on checking the tabu lists for tabu neighborhood operators.

Appendix 11 Claim constraint verification procedure

The output of the model gets verified by several procedures to enhance the model's credibility and to ensure food safety by preventing incorrect schedules that might lead to cross-contamination of allergens, as described in Section 6.3. The pseudo-code in Figure 47 illustrates a verification procedure that checks whether the claim constraints (see Section 2.2.2) are satisfied. This function returns true when all claim constraints are satisfied, and false otherwise.

Verification of claim constraints

```
For k in M do
    For pos in Machine[k].Sequence do
        o_{j,r,i} ← Machine[k].Sequence[pos]
        For c in Claims do
            If o_{j,r,i}.Claim[c] = "certified" then
                For prev = max{0, pos − 1} downto max{0, pos − 2} do
                    o_{j',r',i'}.Claim[c] ← Machine[k].Sequence[prev]
                    If o_{j',r',i'}.Claim[c] = "non − suitable" then
                        Return False
                    EndIf
                End
            EndIf
        End
    End
End
Return True
```

Figure 47 | Pseudo code that verifies whether the claim constraints are satisfied

Appendix 12 Problem instances | configuring eligible production routes

The data of Euroma currently lacks the eligible production routes per job. Therefore, this section explains how we obtain the eligible production routes per production job based on historical data. For further reference, Section 2.1.3 describes production routes and Appendix 3 provides technical information regarding production routes.

At first, whether a production route is eligible for a job depends among others on the job quantity, whether the job needs sieving, and the final package unit of the job. Table 35 presents all the available production routes, including the job requirements for that production route to be eligible, and the corresponding mixing and packaging machines.

Euroma currently does not have a dataset with the minimum and maximum quantity per job per production route. The ERP system only contains a set of possible quantities per job that are linked to one or multiple eligible production routes. However, only some eligible production routes are stored in the dataset, as setting up these production routes in the ERP system is a manual task, which is very time-consuming. Therefore, to obtain the full set of eligible production routes, we extract the job quantities produced per machine over the period 09-10-2020 until 11-03-2021 from the ESA database, where N=4563. We set the minimum and maximum quantity per machine at the 10^{th} and 90^{th} percentile, respectively.

Besides that, routes 17 and 18 only allow one IBC, as the Z409 mixer rotates one single IBC at the time, as described in Section 2.1.2.

Table 35 | Eligible production routes per job requirement

	Job requirements			Mixing			Packaging
Route Nr	Job quantity (kg)	Sieve in route	Final package unit	Filling	Mixer	Discharge unit	Discharge machine
1	[20-199]	Yes	Bag	Manual	Z401	Bag	-
2	[200-1100]	Yes	Bag	IBC	Z402	Bag	-
3	[270-2500]	Yes	Big bag	IBC	Z403	IBC	Z420
4	[270-2500]	Yes	Bag	IBC	Z403	IBC	Z410
5	[270-2500]	Yes	Bag	IBC	Z403	Big bag	Z412
6	[270-2500]	No	Big bag	IBC	Z403	Big bag	-
7	[900-3400]	Yes	Big bag	IBC	Z404	IBC	Z420
8	[900-3400]	Yes	Bag	IBC	Z404	IBC	Z410
9	[900-3400]	Yes	Bag	IBC	Z404	Big bag	Z412
10	[900-3400]	No	Big bag	IBC	Z404	Big bag	-
11	[270-2500]	Yes	Big bag	IBC	Z405	IBC	Z420
12	[270-2500]	Yes	Bag	IBC	Z405	IBC	Z410
13	[270-2500]	Yes	Bag	IBC	Z405	Big bag	Z412
14	[270-2500]	No	Big bag	IBC	Z405	Big bag	-
15	[1700-7500]	Yes	Big bag	IBC/silo	Z407	Big bag	-
16	[1700-7500]	Yes	Big bag	IBC/silo	Z408	Big bag	-
17	[200-1100]	Yes	Big bag	1 IBC	Z409	IBC	Z420
18	[200-1100]	Yes	Bag	1 IBC	Z409	IBC	Z410

For an example of selecting eligible production routes for a job, consider a job with a quantity of 2200kg that does not require sieving and needs a big bag as the final package unit. This job can be produced on the production routes 3, 6, 7, 10, 11, 14, 15, and 16.

Appendix 13 Problem instances | setting the processing times

This section provides an analysis of the processing times of the processes IBC-filling, mixing, and packaging.

IBC-filling

Euroma aims to have a processing time shorter than 45 minutes for every IBC at the IBC-filling station. Therefore, we set the processing time per IBC at the IBC-filling station at 45 minutes.

Mixing

The mixing process consist of (i) filling the mixer, (ii) mixing, (iii) adding liquids manually if required, and (iv) discharging the mixer. Figure 48 shows the filling time per IBC of the 3K, 4.5K, and the 10K mixers. This data was extracted from the ESA IBC transport log over the period 01-01-2020 until 31-12-2020. Together with the process engineers of Euroma, we decide to set the time to fill a mixer with one IBC at 15 minutes, as the majority of the IBCs are expected to complete the filling process within this time in the near future.

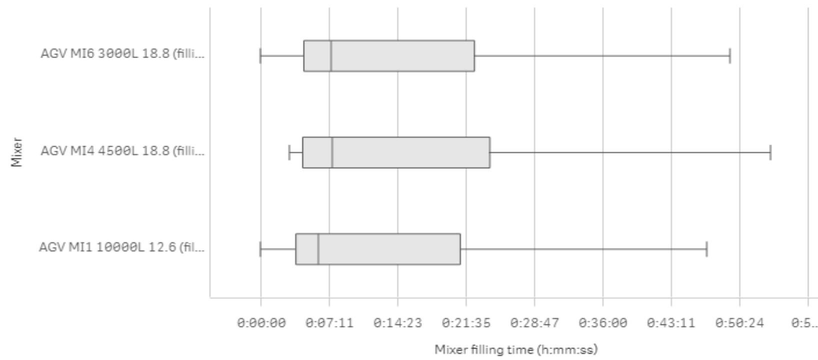


Figure 48 | Time to fill a mixer with one IBC

We set the processing time of the mixing process at 15 minutes, as this is the maximum possible processing time according to the process engineers of Euroma since these processing times are pre-programmed. Besides that, in the case that a mixture requires liquids, we add an extra 15 minutes of processing time for the operator to manually dose the liquids.

We set the time to discharge a mixture from the mixer into a big bag at 20 minutes per big bag. Besides that, Figure 49 shows that discharging the 3K and the 4.5K mixers takes on average 16 minutes per IBC. Together with the process engineers of Euroma, we decide to set the time to discharge a mixer with one IBC at 30 minutes, as the majority of the IBCs are expected to complete the discharging process within this time in the near future. Note that discharging the mixer takes more time than filling the mixer, as discharging the mixer requires additional activities such as sieving and weighing the product.

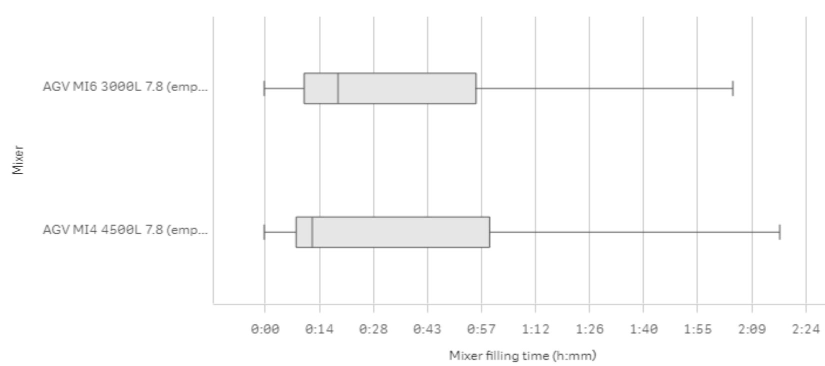


Figure 49 | Time to discharge a mixer with one IBC

Packaging

After discharging the mixtures, some mixtures need packaging. Figure 50 provides a histogram of the average packaging speed in kilograms per minute per job of the Votech (Z410) packaging line. This data was extracted from the MES database over the period 05-01-2020 until 20-05-2021, where N=1124. Together with Euroma, we choose to set the processing speed at 67 bags per hour.

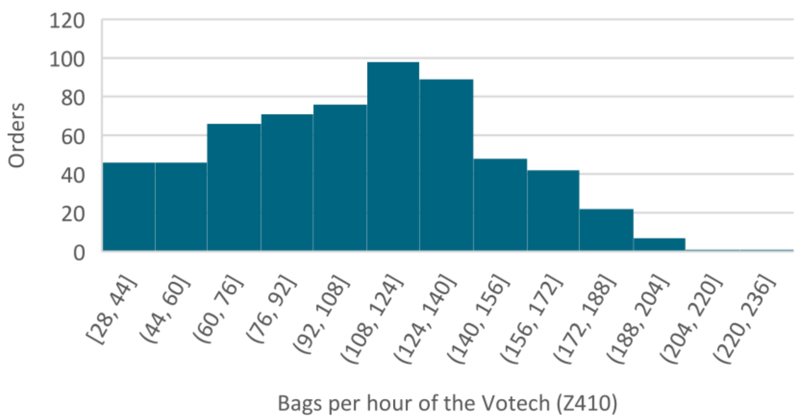


Figure 50 | Processing speed in bags per hour of the Votech (Z410)

Furthermore, Figure 51 provides a histogram of the average packaging speed in kilograms per minute per job of the BTH (Z412) packaging line. This data was extracted from the MES database over the period 05-01-2020 until 20-05-2021, where N=388. Together with Euroma, we choose to set the processing speed at 33 bags per hour.

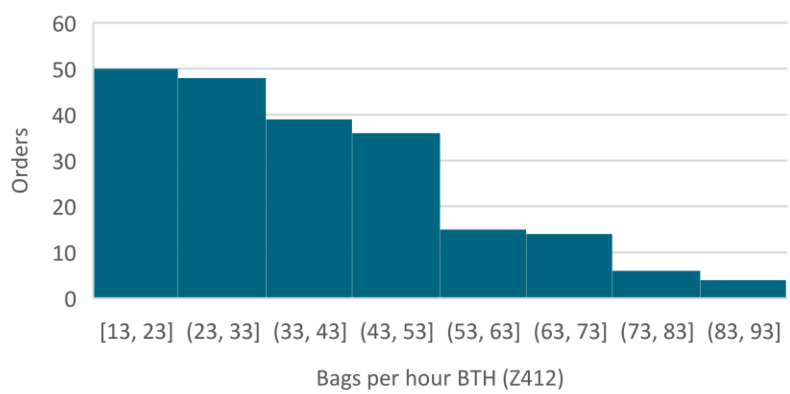


Figure 51 | Processing speed in bags per hour of the BTH (Z412)

The Dinnissen (Z420) has a packaging speed of between 15 and 20 minutes per big bag. We set the speed of this packaging line at 20 minutes per big bag.

Besides that, we set $transp_{k,k'}$, which is the time to transport an IBC or a big bag between any two machines $k \in M_{j,r,i}$ and $k' \in M_{j,r,i}$ at 15 minutes. Moreover, the two IBC-cleaning stations each have a cleaning time of $cleaning^{IBC}$ =16 minutes.

The processing times are validated by the process engineer and the business analyst of Euroma. According to the experts, the processing times that we set seem reasonable.

Appendix 14 Problem instances | configuring the contamination matrix

Recall from Section 2.2.2 that on IBC-filling stations, mixers, and packaging lines, cleaning between two consecutive jobs on the same machine is required when at least one of the following three conditions is applicable: (i) when producing a non-allergen product after an allergen product (containing, e.g., gluten, eggs, or sesame), (ii) when colors of two consecutive products can blend into another color, and (iii) when the raw materials of two products have different physical characteristics (e.g., aroma, particles structure, or stickiness). The data of Euroma currently lacks a contamination matrix. Therefore, this section explains how we obtain the contamination matrix.

Also recall that there are two cleaning types: dry-cleaning and wet-cleaning. Wet-cleaning takes longer than dry-cleaning, as this cleaning type is more intensive. The type of cleaning depends on the colors, physical characters, and allergens of two consecutive products on the same machine. Next, this section explains how to determine the cleaning type.

At first, wet cleaning is always required when sequencing a product that contains different allergens (e.g., gluten, soya, or celery) before a product that does not contain the same allergens. Moreover, tests were done in the laboratory of Euroma to determine which cleaning type is required based on the colors of the products. Table 36 provides the resulting contamination matrix regarding the colors, where "-" refers to no cleaning, "Dry" refers to dry-cleaning, and "Wet" refers to wet-cleaning. A similar contamination matrix was created in the laboratory for the physical characteristics.

Table 36 | Color contamination matrix

From/To	White	White-green	Gray	Green	Light brown	Yellow	Dark yellow	Orange	Red	Brown	Dark brown	Black
White	-	-	-	-	-	-	-	-	-	-	-	-
White-green	-	-	-	-	-	-	-	-	-	-	-	-
Gray	-	-	-	-	-	-	-	-	-	-	-	-
Green	Dry	Dry	Dry	-	-	-	-	-	-	-	-	-
Light brown	Dry	Dry	Dry	-	-	-	-	-	-	-	-	-
Yellow	Dry	Dry	Dry	-	-	-	-	-	-	-	-	-
Dark yellow	Dry	Dry	Dry	-	-	-	-	-	-	-	-	-
Orange	Wet	Wet	Wet	Wet	Dry	Dry	Dry	-	-	-	-	-
Red	Wet	Wet	Wet	Wet	Wet	Wet	Wet	-	-	-	-	-
Brown	Wet	Wet	Wet	Wet	Wet	Dry	Dry	Dry	-	-	-	-
Dark brown	Wet	Wet	Wet	Wet	Wet	Dry	Dry	Dry	Dry	Dry	-	-
Black	Wet	Wet	Wet	Wet	Wet	Wet	Wet	Wet	Wet	Dry	Dry	-

The cleaning duration depends on the cleaning type and the machines. Table 37 reports the cleaning duration per machine and cleaning type according to the control room operators.

Table 37 | Cleaning duration per machine and cleaning type

Cleaning type	IBC Filling	Z401	Z402	Z403	Z404	Z405	Z407	Z408	Z409	Z410	Z412	Z420
Dry cleaning	10	45	45	30	30	30	75	75	-	45	45	45
Wet cleaning	30	75	75	75	75	75	150	150	-	75	75	75

Finally, a script generates a table with the cleaning time between two consecutive jobs per machine based on the (i) allergens information, (ii) color matrix, (iii) physical character matrix, and (iv) the cleaning duration table.

Appendix 15 Detailed results of experiment 1

In Section 5.3.1, we choose to select the weight set $\{14\%, 14\%, 14\%, 28\%, 30\%\}$ based on problem instance 3. To evaluate whether this weight set is also appropriate for all other problem instances, this section provides the results of 10 replications per problem instance, of which the standard deviation is reported between brackets.

Table 38 | Evaluating the selected weight set on all problem instances

Instance	C_{max}	T_{tot}	CT_{tot}	BT_{avg}	IBC_{max}	Feasible	CPU
1	3:21:43:30 (2:55:33)	0	3:23:13:00 (4:19:51)	2:07:03 (25:24)	42.6 (4.2)	100%	85.0 (1.1)
2	3:12:30:30 (2:05:33)	0	3:03:10:30 (4:34:35)	2:05:16 (22:03)	43.0 (3.2)	100%	83.9 (0.9)
3	5:05:33:00 (3:14:37)	0	6:19:06:30 (5:28:39)	3:41:32 (28:35)	49.0 (4.2)	100%	137.0 (2.1)
4	4:19:55:48 (2:40:26)	0	5:08:40:00 (5:26:46)	3:06:36 (29:17)	47.0 (3.3)	100%	131.3 (0.9)
5	6:09:54:18 (2:59:06)	0	9:11:57:30 (6:50:26)	4:10:02 (33:27)	51.6 (4.6)	100%	182.9 (1.5)
6	6:13:27:42 (3:21:55)	0	9:07:37:00 (6:21:39)	4:15:01 (35:25)	53.5 (4.6)	100%	192.8 (1.0)

From the results in Table 38, we observe that the selected weight set finds feasible solutions for all problem instances over all replications. Therefore, we select this weight set for all other experiments.

Appendix 16 Detailed results of experiment 2

Table 39 provides the detailed experimental results of Exp2 on the alternative model configurations per problem instance, as described in Section 5.2.1. For every problem instance and scenario, this experiment performs 10 replications. Section 5.3.2 provides a summary and an analysis of these results.

Table 39 | Detailed results of experiment 2

Ins	Scn	Configuration	Obj	C_{max}	T_{tot}	CT_{tot}	BT_{avg}	IBC_{max}	Feasible	CPU
1	1	RCH, SIH, RN	54627 53434	10:00:58:42 (15:02:46)	39:05:13:36 (7:13:14:28)	13:06:56:30 (6:13:42)	3:17:42:40 (6:47:56)	355.9 (18.1)	0%	150.0 (0.0)
	2	RCH, SIH, SN	55894 52767	10:03:51:48 (11:05:53)	38:11:23:36 (6:20:15:45)	13:02:50:30 (7:28:45)	3:16:03:09 (3:24:50)	357.9 (15.5)	0%	150.0 (0.0)
	3	RCH, SA, RN	57035 6994	3:21:07:30 (2:47:58)	0	3:23:48:30 (4:02:11)	2:19:47 (25:09)	40.5 (2.9)	100%	83.2 (1.5)
	4	RCH, SA, SN	56792 7067	3:21:01:30 (1:47:56)	0	4:03:17:30 (4:06:17)	2:14:12 (24:51)	41.0 (2.4)	100%	96.2 (4.0)
	5	RCH, SATL, RN	55876 7118	3:22:49:30 (2:05:51)	0	4:03:10:00 (6:15:52)	2:19:09 (28:53)	41.1 (2.8)	100%	67.6 (1.6)
	6	RCH, SATL, SN	57965 7431	3:22:03:00 (2:10:09)	0	4:07:59:00 (5:40:58)	3:25:27 (33:10)	48.8 (5.0)	100%	70.7 (0.7)
	7	NEH, SIH, RN	20254 20254	5:11:01:18 (7:47:45)	13:12:45:00 (1:09:26:17)	7:05:00:00 (6:33:08)	17:21:28 (2:13:29)	114.2 (20.8)	0%	8.1 (0.2)
	8	NEH, SIH, SN	20878 20878	5:09:58:06 (4:52:41)	14:05:07:00 (1:20:58:14)	7:08:06:00 (8:45:59)	18:30:22 (3:11:14)	116.9 (17.1)	0%	8.8 (0.1)
	9	NEH, SA, RN	20314 6982	3:21:43:30 (2:55:33)	0	3:23:13:00 (4:19:51)	2:07:03 (25:24)	42.6 (4.2)	100%	85.0 (1.1)
	10	NEH, SA, SN	20294 7018	3:22:00:00 (2:21:34)	0	4:01:25:00 (5:30:52)	2:12:17 (20:34)	42.2 (3.2)	100%	104.4 (12.8)
	11	NEH, SATL, RN	20110 7228	4:01:16:00 (3:08:15)	0	4:04:43:30 (4:11:15)	2:22:38 (30:52)	40.3 (3.4)	100%	73.1 (1.2)
	12	NEH, SATL, SN	20524 7473	3:22:47:30 (1:44:35)	0	4:10:04:00 (2:51:59)	3:26:39 (32:23)	46.5 (3.3)	100%	112.6 (8.5)
2	1	RCH, SIH, RN	55017 51328	10:03:01:48 (16:39:58)	34:00:12:12 (4:16:13:36)	11:09:52:00 (4:44:08)	3:15:54:52 (4:28:44)	353.1 (19.0)	0%	150.0 (0.0)
	2	RCH, SIH, SN	51397 47529	10:02:35:48 (21:37:12)	28:01:58:24 (3:04:02:27)	11:03:01:00 (7:59:40)	3:14:14:19 (6:36:33)	344.9 (18.7)	0%	150.0 (0.0)
	3	RCH, SA, RN	55276 6409	3:14:30:00 (1:36:36)	0	3:02:45:00 (4:49:03)	2:01:21 (20:02)	43.7 (4.0)	100%	81.9 (1.0)
	4	RCH, SA, SN	55988 6440	3:13:54:00 (3:28:56)	0	3:04:04:00 (4:09:39)	2:13:11 (30:26)	46.6 (4.1)	100%	103.1 (9.1)
	5	RCH, SATL, RN	54945 6613	3:15:07:30 (2:22:57)	0	3:08:17:30 (6:02:48)	2:13:20 (33:31)	45.5 (4.1)	100%	70.2 (2.5)
	6	RCH, SATL, SN	56682 6809	3:14:03:30 (2:23:11)	0	3:08:50:00 (3:10:20)	3:33:43 (32:29)	52.8 (3.9)	100%	69.0 (0.6)
	7	NEH, SIH, RN	18602 18602	5:00:53:54 (7:12:52)	10:06:58:06 (1:21:03:35)	6:05:09:30 (7:35:34)	18:52:28 (2:04:49)	146.8 (14.3)	0%	8.0 (0.1)
	8	NEH, SIH, SN	18461 18461	4:23:25:06 (9:28:35)	10:03:51:48 (2:14:28:24)	6:06:26:30 (5:09:49)	17:54:12 (2:09:01)	137.5 (13.9)	0%	8.9 (0.5)
	9	NEH, SA, RN	17937 6385	3:12:30:30 (2:05:33)	0	3:03:10:30 (4:34:35)	2:05:16 (22:03)	43.0 (3.2)	100%	83.9 (0.9)
	10	NEH, SA, SN	17560 6405	3:14:09:00 (2:11:46)	0	3:02:55:30 (4:37:27)	2:03:34 (29:04)	44.2 (5.2)	100%	90.0 (1.8)
	11	NEH, SATL, RN	19134 6641	3:16:00:30 (2:54:29)	0	3:07:38:30 (3:30:04)	2:19:09 (30:21)	44.7 (2.2)	100%	71.8 (0.9)
	12	NEH, SATL, SN	18554 6844	3:13:51:00 (2:52:22)	0	3:10:59:30 (3:55:49)	3:27:31 (34:36)	52.1 (5.2)	90%	92.8 (16.4)

3	1	RCH, SIH, RN	68416 66100	18:08:10:36 (1:06:00:34)	65:07:53:36 (6:09:59:50)	18:03:01:00 (10:03:13)	6:11:57:47 (8:16:47)	542.2 (30.3)	0%	150.0 (0.0)
	2	RCH, SIH, SN	69777 64300	17:18:12:18 (22:27:32)	59:00:46:00 (9:14:08:31)	17:23:27:00 (9:18:05)	6:13:40:41 (10:07:16)	532.2 (28.9)	0%	150.0 (0.0)
	3	RCH, SA, RN	66491 7390	5:05:12:18 (1:42:17)	0	6:19:41:30 (5:50:27)	3:50:41 (31:55)	49.4 (3.3)	100%	133.1 (2.3)
	4	RCH, SA, SN	65310 7285	5:04:48:54 (2:02:31)	0	6:16:33:00 (9:04:43)	3:33:56 (19:01)	49.5 (2.2)	100%	144.6 (2.7)
	5	RCH, SATL, RN	67701 7437	5:05:42:42 (4:04:09)	0	6:22:14:30 (8:20:48)	4:00:25 (28:35)	50.3 (5.5)	100%	125.4 (10.1)
	6	RCH, SATL, SN	69528 7751	5:04:27:00 (2:45:11)	0	7:02:49:00 (3:41:52)	5:26:35 (32:06)	59.7 (6.9)	60%	110.9 (1.2)
	7	NEH, SIH, RN	21503 21488	7:05:59:42 (8:41:42)	21:16:55:36 (3:10:47:42)	10:16:42:00 (4:15:57)	1:01:18:11 (2:47:53)	170.5 (21.2)	0%	15.6 (0.3)
	8	NEH, SIH, SN	22572 22572	7:22:27:36 (13:48:21)	24:01:36:18 (4:13:23:02)	10:10:58:30 (6:06:39)	1:01:46:58 (2:30:09)	157.8 (21.1)	0%	16.9 (0.1)
	9	NEH, SA, RN	21809 7370	5:05:33:00 (3:14:37)	0	6:19:06:30 (5:28:39)	3:41:32 (28:35)	49.0 (4.2)	100%	137.0 (2.1)
	10	NEH, SA, SN	21510 7413	5:05:22:00 (3:14:40)	0	6:21:41:00 (4:54:14)	3:44:27 (19:35)	47.5 (3.4)	100%	144.3 (0.9)
	11	NEH, SATL, RN	22339 7493	5:06:56:30 (2:46:46)	0	6:21:46:30 (4:54:04)	4:00:16 (22:05)	51.0 (2.4)	100%	125.9 (4.7)
	12	NEH, SATL, SN	22087 7754	5:04:31:42 (2:59:24)	0	7:03:09:30 (3:15:55)	5:28:23 (38:29)	60.5 (5.6)	60%	126.1 (1.8)
4	1	RCH, SIH, RN	55288 54384	17:06:52:18 (21:39:04)	46:20:43:54 (10:19:30:18)	17:17:17:30 (8:18:16)	5:05:51:38 (11:34:11)	468.0 (32.9)	0%	131.4 (44.7)
	2	RCH, SIH, SN	56307 53514	17:04:20:54 (21:03:37)	44:13:05:12 (4:13:34:49)	17:15:17:30 (7:49:42)	5:04:35:40 (6:12:08)	473.6 (29.5)	0%	150.0 (0.0)
	3	RCH, SA, RN	56729 6561	4:19:55:12 (2:26:05)	0	5:10:38:30 (4:04:37)	3:10:30 (29:55)	48.6 (3.2)	100%	128.8 (0.8)
	4	RCH, SA, SN	56689 6604	4:20:36:24 (1:11:39)	0	5:10:36:30 (5:08:26)	3:17:47 (23:13)	51.1 (6.7)	90%	137.9 (1.9)
	5	RCH, SATL, RN	59289 6652	4:22:50:54 (2:19:05)	0	5:10:43:30 (6:41:01)	3:13:53 (17:55)	47.2 (4.0)	100%	121.3 (9.1)
	6	RCH, SATL, SN	55633 6941	4:20:51:18 (2:03:53)	0	5:19:14:00 (7:25:52)	4:31:59 (40:47)	57.1 (3.1)	90%	106.2 (5.1)
	7	NEH, SIH, RN	21625 21625	7:05:49:42 (13:44:38)	23:22:05:24 (1:17:51:14)	8:14:34:30 (11:50:54)	23:29:04 (3:00:14)	162.1 (16.6)	0%	14.5 (0.2)
	8	NEH, SIH, SN	21135 21131	7:06:27:48 (14:51:35)	22:01:53:00 (2:12:11:24)	8:12:46:30 (9:39:44)	1:01:26:11 (3:07:49)	168.8 (17.0)	0%	15.6 (0.1)
	9	NEH, SA, RN	21350 6533	4:19:55:48 (2:40:26)	0	5:08:40:00 (5:26:46)	3:06:36 (29:17)	47.0 (3.3)	100%	131.3 (0.9)
	10	NEH, SA, SN	21461 6560	4:21:49:18 (2:15:33)	0	5:08:17:00 (5:35:49)	3:11:20 (17:41)	45.6 (4.3)	100%	137.7 (5.2)
	11	NEH, SATL, RN	21441 6752	4:21:13:36 (3:19:13)	0	5:16:13:30 (7:24:23)	3:35:24 (28:19)	50.0 (4.4)	100%	128.8 (16.6)
	12	NEH, SATL, SN	22037 6965	4:20:54:42 (2:49:11)	0	5:21:08:30 (5:09:44)	4:23:22 (26:41)	54.3 (3.3)	100%	121.3 (1.1)
5	1	RCH, SIH, RN	85093 82725	23:12:51:36 (1:11:34:54)	102:01:34:30 (11:01:58:51)	24:02:36:30 (7:54:53)	7:23:57:46 (12:27:04)	668.2 (25.9)	0%	150.0 (0.0)
	2	RCH, SIH, SN	85950 82360	23:02:28:24 (1:16:43:35)	97:22:04:06 (9:00:18:16)	24:06:07:30 (10:32:22)	8:03:52:20 (15:12:08)	660.9 (19.5)	0%	150.0 (0.0)
	3	RCH, SA, RN	83074 7519	6:11:42:24 (3:23:47)	0	9:09:27:30 (6:56:15)	4:07:10 (21:38)	54.2 (6.4)	80%	174.7 (1.2)
	4	RCH, SA, SN	80915 7462	6:09:35:06 (2:44:55)	0	9:07:52:30 (6:23:29)	4:12:35 (24:19)	52.4 (4.5)	90%	188.1 (2.7)

5	5	RCH, SATL, RN	88901 7677	6:12:06:06 (3:38:54)	0	9:16:08:00 (9:59:04)	4:25:26 (24:05)	54.4 (6.8)	90%	149.6 (1.6)
	6	RCH, SATL, SN	84338 7905	6:08:51:30 (3:20:30)	0	9:21:06:30 (7:53:55)	5:49:22 (28:35)	62.8 (3.0)	30%	154.4 (9.7)
	7	NEH, SIH, RN	22029 22029	9:14:12:00 (14:45:07)	25:00:13:42 (5:02:59:00)	11:22:37:00 (15:08:02)	1:07:43:39 (5:19:05)	237.5 (36.6)	0%	22.9 (0.3)
	8	NEH, SIH, SN	23606 23601	10:00:36:54 (20:19:28)	28:22:37:42 (4:18:01:43)	11:20:00:30 (11:54:21)	1:10:17:30 (3:16:41)	227.7 (44.9)	0%	25.3 (0.3)
	9	NEH, SA, RN	24519 7548	6:09:54:18 (2:59:06)	0	9:11:57:30 (6:50:26)	4:10:02 (33:27)	51.6 (4.6)	100%	182.9 (1.5)
	10	NEH, SA, SN	23420 7521	6:11:28:24 (2:53:11)	0	9:10:45:30 (7:31:13)	4:03:47 (25:35)	52.5 (5.2)	90%	189.8 (6.6)
	11	NEH, SATL, RN	23999 7770	6:12:55:54 (2:56:05)	0	9:20:03:30 (7:39:15)	4:21:44 (21:11)	55.2 (4.4)	90%	158.5 (3.3)
	12	NEH, SATL, SN	23997 7941	6:10:12:18 (2:59:19)	0	9:20:48:00 (6:29:30)	5:55:03 (27:24)	65.3 (3.7)	10%	162.8 (5.0)
6	1	RCH, SIH, RN	102553 100442	22:07:02:18 (2:01:01:06)	163:05:09:12 (14:05:27:38)	23:21:12:30 (8:43:58)	8:12:35:09 (17:35:02)	690.4 (35.4)	0%	150.0 (0.0)
	2	RCH, SIH, SN	98686 94880	22:01:59:30 (1:14:43:43)	150:02:01:42 (12:10:10:43)	23:10:11:30 (11:52:09)	8:02:14:32 (12:18:20)	680.4 (24.3)	0%	150.0 (0.0)
	3	RCH, SA, RN	99344 7580	6:15:09:00 (2:31:00)	0	9:09:05:00 (7:45:00)	4:07:28 (25:37)	50.4 (3.0)	100%	185.5 (3.3)
	4	RCH, SA, SN	103542 7615	6:15:09:30 (1:50:29)	0	9:12:54:30 (5:48:34)	4:12:41 (18:58)	50.4 (4.6)	100%	208.2 (8.8)
	5	RCH, SATL, RN	101289 7758	6:15:01:36 (3:58:14)	0	9:15:07:30 (7:11:51)	4:39:14 (21:22)	55.6 (7.9)	80%	157.8 (2.5)
	6	RCH, SATL, SN	100923 7975	6:12:33:42 (1:47:36)	0	9:18:25:00 (6:26:55)	6:02:58 (37:41)	63.6 (5.1)	30%	184.8 (32.4)
	7	NEH, SIH, RN	24317 24311	9:06:19:36 (15:16:56)	34:00:10:54 (6:21:14:53)	11:17:02:30 (15:53:16)	1:07:22:29 (2:13:22)	222.1 (17.3)	0%	24.9 (0.3)
	8	NEH, SIH, SN	22527 22523	8:23:46:48 (18:32:46)	29:01:23:18 (5:10:29:56)	11:09:59:30 (7:42:08)	1:07:21:07 (3:04:32)	230.9 (33.9)	0%	28.2 (1.4)
	9	NEH, SA, RN	25127 7568	6:13:27:42 (3:21:55)	0	9:07:37:00 (6:21:39)	4:15:01 (35:25)	53.5 (4.6)	100%	192.8 (1.0)
	10	NEH, SA, SN	26006 7539	6:13:03:00 (2:45:52)	0	9:05:22:00 (7:13:32)	4:18:57 (29:23)	54.1 (4.7)	90%	204.1 (1.5)
	11	NEH, SATL, RN	25050 7822	6:17:13:30 (2:42:24)	0	9:17:11:00 (8:06:07)	4:40:38 (32:12)	56.4 (5.4)	60%	165.8 (1.6)
	12	NEH, SATL, SN	24749 8054	6:13:29:00 (3:41:26)	0	9:23:02:30 (6:33:44)	6:02:11 (38:21)	65.0 (4.6)	20%	172.5 (4.9)

Appendix 17 Detailed results of experiment 3

Table 40 provides the detailed experimental results of Exp3, which evaluates the effect of optimizing the schedules of the stages simultaneously compared to the current situation, which optimizes the stages separately, as described in Section 5.2.2. For every problem instance and scenario, this experiment performs 25 replications. Section 5.4.1 provides a summary and an analysis of these results.

Table 40 | Detailed results of experiment 3

Instance	Scenario	C_{max}	T_{tot}	CT_{tot}	BT_{avg}	IBC_{max}	Feasible
1	Current situation	4:19:28:09 (5:50:28)	0	4:21:23:48 (5:28:53)	1:45:41 (32:11)	38.9 (4.5)	100%
	Simultaneous optimization	4:07:54:36 (29:13)	0	3:17:33:48 (3:22:02)	2:52:35 (27:34)	47.3 (4.3)	100%
2	Current situation	4:02:24:14 (6:33:56)	15:20:40 (19:23:35)	4:03:32:12 (4:02:06)	2:24:59 (31:51)	42.5 (4.9)	100%
	Simultaneous optimization	3:14:57:31 (1:55:25)	0	2:23:29:36 (3:05:48)	3:28:55 (28:29)	50.6 (3.6)	100%
3	Current situation	6:14:34:19 (2:16:29)	1:17:43:57 (1:11:05:27)	6:17:29:12 (3:44:30)	6:17:28 (1:13:58)	59.7 (8.4)	48%
	Simultaneous optimization	6:14:46:26 (1:47:45)	0)	6:02:16:00 (5:48:18)	5:42:58 (44:40)	57.8 (5.6)	68%
4	Current situation	6:02:18:16 (3:07:05)	1:01:55 (5:09:36)	6:05:31:48 (3:48:40)	3:29:39 (44:22)	49.4 (5.3)	100%
	Simultaneous optimization	6:00:18:21 (2:30:39)	0	5:08:02:48 (5:53:42)	4:47:54 (35:45)	56.9 (5.3)	72%
5	Current situation	8:03:59:12 (2:44:52)	3:55:00 (10:51:49)	8:03:59:24 (6:17:01)	7:55:15 (1:01:03)	71.0 (6.5)	4%
	Simultaneous optimization	8:06:40:38 (5:31:27)	0	8:01:08:48 (6:50:56)	7:01:05 (1:01:33)	65.1 (6.1)	24%
6	Current situation	8:05:56:12 (2:14:07)	8:21:44:02 (3:22:05:17)	8:03:45:36 (5:53:26)	8:20:20 (1:19:03)	69.2 (6.4)	12%
	Simultaneous optimization	8:10:41:48 (3:50:35)	3:42:48 (10:25:43)	8:01:18:00 (9:24:02)	7:17:50 (51:50)	66.9 (6.7)	20%

Appendix 18 Detailed results of experiment 4

Table 41 provides the detailed experimental results of Exp4, which compares the ability to allow (i) only the default production route and (ii) changing the production route to any other eligible production route. We refer to the first scenario as “Model (default routes)” and to the second scenario as “Model (eligible routes)”. For every problem instance and scenario, this experiment performs 25 replications. Section 5.4.1 provides a summary and an analysis of these results.

Table 41 | Detailed results of experiment 4

Instance	Scenario	C_{max}	T_{tot}	CT_{tot}	BT_{avg}	IBC_{max}	Feasible
1	Model (default route)	4:07:54:36 (29:13)	0	3:17:33:48 (3:22:02)	2:52:35 (27:34)	47.3 (4.3)	100%
	Model (eligible routes)	3:22:08:00 (2:28:58)	0	3:23:49:00 (4:21:03)	2:09:02 (24:04)	41.2 (3.9)	100%
2	Model (default route)	3:14:57:31 (1:55:25)	0	2:23:29:36 (3:05:48)	3:28:55 (28:29)	50.6 (3.6)	100%
	Model (eligible routes)	3:13:09:24 (1:50:37)	0	3:02:56:48 (4:13:17)	2:06:30 (25:04)	43.3 (3.9)	100%
3	Model (default route)	6:14:46:26 (1:47:45)	0	6:02:16:00 (5:48:18)	5:42:58 (44:40)	57.8 (5.6)	68%
	Model (eligible routes)	5:05:05:48 (2:24:30)	0	6:18:23:24 (6:40:45)	3:59:05 (45:07)	49.7 (4.8)	100%
4	Model (default route)	6:00:18:21 (2:30:39)	0	5:08:02:48 (5:53:42)	4:47:54 (35:45)	56.9 (5.3)	72%
	Model (eligible routes)	4:20:33:33 (2:15:14)	0	5:09:34:24 (6:44:27)	3:12:33 (31:11)	49.0 (5.1)	96%
5	Model (default route)	8:06:40:38 (5:31:27)	0	8:01:08:48 (6:50:56)	7:01:05 (1:01:33)	65.1 (6.1)	24%
	Model (eligible routes)	6:08:44:38 (2:50:49)	0	9:07:49:48 (8:59:09)	4:11:00 (33:01)	52.1 (5.4)	92%
6	Model (default route)	8:10:41:48 (3:50:35)	0:03:42:48 (10:25:43)	8:01:18:00 (9:24:02)	7:17:50 (51:50)	66.9 (6.7)	20%
	Model (eligible routes)	6:13:41:04 (2:39:55)	0	9:07:03:24 (6:13:28)	4:18:43 (32:15)	53.6 (5.1)	92%

Appendix 19 Statistical results on the evaluation of the model in practice

Section 5.4.2 tests if there is a significant reduction in the cleaning time after implementing the model in practice. To test this, we perform a two-sample t-test in which we assume unequal variances. Moreover, we assume that all samples are independent identically distributed. We set alpha at 0.5%, resulting in a p-value of $5.23662*10^{-14}$. Note that the p-value is smaller than alpha. Therefore, we reject the null hypotheses of equal average cleaning times per job before- and after implementation. The corresponding statistical results are in Table 42.

t-Test: Two-sample assuming unequal variances

	Cleaning time per job (minutes)	
	Before implementation	***After implementation***
Mean	30.95389871	19.03470025
Variance	21.79836231	12.77060135
Observations	31	21
Hypothesized Mean Difference	0	
df	49	
t Stat	10.40870121	
P(T<=t) one-tail	2.61831E-14	
t Critical one-tail	2.40489176	
P(T<=t) two-tail	5.23662E-14	
t Critical two-tail	2.679951974	

Table 42 | Statistical results of the implementation of the model in practice